



National Library  
of Canada

Bibliothèque nationale  
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada  
K1A 0N4

## NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

## AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

UNIVERSITY OF ALBERTA

THREE-DIMENSIONAL RECONSTRUCTION OF SKULL  
FROM MAGNETIC RESONANCE IMAGES

BY

FAL-HUNG CHAN

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

FALL 1990



National Library  
of Canada

Bibliothèque nationale  
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada  
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

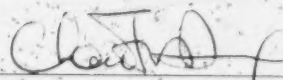
ISBN 0-315-65083-4

UNIVERSITY OF ALBERTA  
RELEASE FORM

NAME OF AUTHOR: Fai-Hung Chan  
TITLE OF THESIS: Three-Dimensional Reconstruction of Skull From  
Magnetic Resonance Images  
DEGREE: Master of Science  
YEARS THIS DEGREE GRANTED: 1990

PERMISSION IS HEREBY GRANTED TO THE UNIVERSITY OF ALBERTA  
LIBRARY TO REPRODUCE SINGLE COPIES OF THIS THESIS AND TO LEND  
OR SELL SUCH COPIES FOR PRIVATE, SCHOLARLY OR SCIENTIFIC  
RESEARCH PURPOSES ONLY.

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER  
THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR  
OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMIS-  
SION.

  
(Student's Signature)

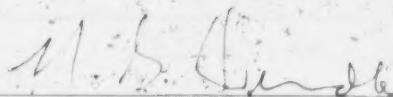
2608, Block K  
Allway Gardens  
Tsuen Wan  
Hong Kong

Date:

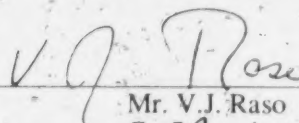
UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

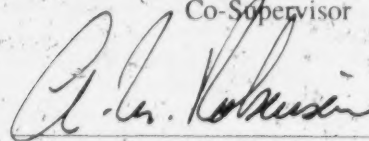
THE UNDERSIGNED CERTIFY THAT THEY HAVE READ, AND RECOMMEND  
TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH FOR  
ACCEPTANCE, A THESIS ENTITLED **THREE-DIMENSIONAL  
RECONSTRUCTION OF SKULL FROM MAGNETIC RESONANCE IMAGES**  
SUBMITTED BY **FAI-HUNG CHAN** IN PARTIAL FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.



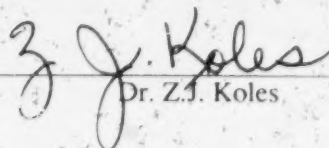
Dr. N.G. Durdle  
Co-Supervisor



Mr. V.J. Raso  
Co-Supervisor



Dr. A.M. Robinson



Dr. Z.J. Koles

Date:

## ABSTRACT

Scaphocephaly refers to the premature closure of the sagittal suture resulting in a long and narrow head. Since it does not significantly affect brain growth, the single reason for surgical intervention is cosmetic improvement. The planning and evaluation of these surgical operations depends on radiological imaging but the three-dimensional structures are not fully represented by the planar format of magnetic resonance imaging. A three-dimensional reconstruction software application is developed to provide a tool for evaluation and visualization of the three-dimensional bony architecture of the skull.

The three-dimensional reconstruction application is implemented by two programs: 'HEAD' and '3D'. The program 'HEAD' performs boundary extraction from a set of magnetic resonance images and provides filtering operations to enhance the quality of the image for the extraction step. The program '3D' uses the boundary data extracted by 'HEAD' to create a three-dimensional model and displays the model using wire-frame display mode with or without hidden surface removal. The application runs on a desktop computer and can be used in a clinical environment to evaluate the effectiveness of the surgical treatment to children with scaphocephaly.

## ACKNOWLEDGEMENTS

I wish to express my profound gratitude to my supervisors, Dr. Nelson G. Durdle and V. James Raso, for their efforts in guiding me through the process of writing and producing this thesis. I would like to thank for their patience, their guidance and their encouragement throughout the work. Thank is due to Mr. Doug Hill for his valuable suggestions and assistance in the development of the software application. I also acknowledge Dr. Zoly Koles, Mr. Joseph DeAlmeida and Mr. Allen Au for their valuable assistance in Macintosh programming.

Lastly, but not the least, I wish to express my appreciation to my parents. Without their foremost patience, continuous encouragement, and support, the completion of this thesis would not have been possible.

Financial support was given by the Department of Electrical Engineering, University of Alberta and the Rehabilitation Engineering Department, Glenrose Rehabilitation Hospital.

## TABLE OF CONTENTS

| CHAPTER                                                    | PAGE |
|------------------------------------------------------------|------|
| 1. INTRODUCTION .....                                      | 1    |
| References .....                                           | 5    |
| 2. BRIEF REVIEW OF MAGNETIC RESONANCE IMAGING .....        | 6    |
| 2.1 Background .....                                       | 6    |
| 2.2 Scanning Sequence .....                                | 9    |
| 2.3 Spatial Errors of Magnetic Resonance Images .....      | 13   |
| 2.4 Magnetic Resonance Images of Scaphocephalic Head ..... | 17   |
| References .....                                           | 20   |
| 3. IMAGE PROCESSING TECHNIQUES FOR MAGNETIC RESONANCE      |      |
| IMAGES OF THE SKULL .....                                  | 21   |
| 3.1 Scope of the Software Application .....                | 21   |
| 3.2 Development Environment .....                          | 21   |
| 3.3 Input File Requirements .....                          | 23   |
| 3.4 Skull Boundary Extraction .....                        | 24   |
| 3.5 Skull Boundary Representation .....                    | 42   |
| 3.6 Image Enhancement by Filtering .....                   | 43   |
| 3.7 Boundary Data File .....                               | 51   |
| 3.8 Three Dimensional Reconstruction Process .....         | 54   |
| 3.9 Hidden Surface Removal .....                           | 56   |
| 3.10 Model Display .....                                   | 56   |
| 3.11 Conclusion .....                                      | 57   |
| References .....                                           | 59   |

|                                                     |     |
|-----------------------------------------------------|-----|
| 4. SOFTWARE APPLICATION .....                       | 61  |
| 4.1 INTRODUCTION .....                              | 61  |
| 4.2 'HEAD' .....                                    | 64  |
| 4.2.1 Description of software routines .....        | 65  |
| 4.2.2 Menu Command .....                            | 83  |
| 4.3 '3D' .....                                      | 88  |
| 4.3.1 Description of software routines .....        | 89  |
| 4.3.2 Menu Command .....                            | 97  |
| References .....                                    | 100 |
| 5. SAMPLE APPLICATION SESSION .....                 | 101 |
| 5.1 'HEAD' .....                                    | 101 |
| 5.2 '3D' .....                                      | 106 |
| 6. CONCLUSION AND RECOMMENDATION FOR FUTURE WORK .. | 113 |
| 6.1 Conclusion .....                                | 113 |
| 6.2 Limitations .....                               | 114 |
| 6.3 Recommendation for future work .....            | 115 |
| References .....                                    | 117 |

## LIST OF TABLES

|           |                                                               |    |
|-----------|---------------------------------------------------------------|----|
| Table 2.1 | Percentage errors of gridline measurements .....              | 15 |
| Table 3.1 | Two common gradient operators .....                           | 31 |
| Table 3.2 | Two different weighting functions of Laplacian operator ..... | 35 |
| Table 3.3 | Unsharp mask .....                                            | 47 |

## LIST OF FIGURES

|             |                                                             |    |
|-------------|-------------------------------------------------------------|----|
| Figure 1.1  | Input data Set .....                                        | 3  |
| Figure 2.1  | The spin echo pulse sequence .....                          | 10 |
| Figure 2.2  | The inversion recovery pulse sequence .....                 | 12 |
| Figure 2.3  | Test image for error measurement .....                      | 14 |
| Figure 2.4  | Magnetic resonance image 1 of scaphocephalic head .....     | 18 |
| Figure 2.5  | Magnetic resonance image 2 of scaphocephalic head .....     | 19 |
| Figure 3.1  | Flow diagram of three-dimensional reconstruction .....      | 22 |
| Figure 3.2  | Result of thresholding to magnetic resonance image 1 .....  | 26 |
| Figure 3.3  | Results of thresholding to magnetic resonance image 2 ..... | 27 |
| Figure 3.4  | Test images of 8 pixel line width .....                     | 35 |
| Figure 3.5  | Performance measurements of $r$ for ideal edges .....       | 36 |
| Figure 3.6  | Performance measurements of $r$ for ramp edges .....        | 36 |
| Figure 3.7  | Performance measurements of $R$ for ideal edges .....       | 37 |
| Figure 3.8  | Performance measurements of $R$ for ramp edges .....        | 37 |
| Figure 3.9  | Contour tracing window .....                                | 40 |
| Figure 3.10 | Example of contour tracing .....                            | 41 |
| Figure 3.11 | Fourier transform of skull image.....                       | 48 |
| Figure 3.12 | Results of ideal low-pass filtering to skull image .....    | 49 |
| Figure 3.13 | Example of image smoothing by neighbourhood averaging ..... | 50 |
| Figure 3.14 | Test image for sharpening operation .....                   | 52 |
| Figure 3.15 | Example of image sharpening by unsharp masking .....        | 53 |
| Figure 3.16 | Three-dimensional model array .....                         | 55 |
| Figure 3.17 | A data cell .....                                           | 55 |
| Figure 3.18 | '3D' display window .....                                   | 58 |
| Figure 4.1  | Main event loop of Macintosh programs .....                 | 61 |

|             |                                            |     |
|-------------|--------------------------------------------|-----|
| Figure 4.2  | Structure chart format .....               | 63  |
| Figure 4.3  | Structure chart of 'HEAD' .....            | 65  |
| Figure 4.4  | Structure chart of HandleEvents .....      | 69  |
| Figure 4.5  | Structure chart of DoMenuEvent .....       | 72  |
| Figure 4.6  | Structure chart of DoDigitize .....        | 74  |
| Figure 4.7  | Structure chart of GetFile .....           | 76  |
| Figure 4.8  | Structure chart of DoClose .....           | 76  |
| Figure 4.9  | Structure chart of RevertToSaved .....     | 77  |
| Figure 4.10 | Structure chart of DoSaveEdge .....        | 79  |
| Figure 4.11 | Structure chart of ProcessDig .....        | 79  |
| Figure 4.12 | Structure chart of DoOutLine .....         | 80  |
| Figure 4.13 | Structure chart of DoEdge .....            | 80  |
| Figure 4.14 | Structure chart of ContourTracing .....    | 81  |
| Figure 4.15 | Structure chart of DoAutoTrace .....       | 81  |
| Figure 4.16 | Structure chart of AutoProcess .....       | 82  |
| Figure 4.17 | Structure chart of File Menu .....         | 83  |
| Figure 4.18 | Structure chart of Edit Menu .....         | 85  |
| Figure 4.19 | Structure chart of Functions Menu .....    | 86  |
| Figure 4.20 | Structure chart of '3D' .....              | 89  |
| Figure 4.21 | Structure chart of DoEvent .....           | 92  |
| Figure 4.22 | Structure chart of DoMenuCommand .....     | 93  |
| Figure 4.23 | Structure chart of DoUpdate .....          | 96  |
| Figure 4.24 | Structure chart of DisplayModel .....      | 97  |
| Figure 4.25 | Structure chart of File Menu .....         | 98  |
| Figure 4.26 | Structure chart of Display Menu .....      | 99  |
| Figure 5.1  | Dialog boxes for 'New Set' Menu Item ..... | 103 |
| Figure 5.2  | Dialog boxes for 'New Set' Menu Item ..... | 104 |

|            |                                                    |     |
|------------|----------------------------------------------------|-----|
| Figure 5.3 | Start-up window after reading-in data .....        | 105 |
| Figure 5.4 | Result of applying Smooth and Edge Detection ..... | 105 |
| Figure 5.5 | Preliminary '3D' display .....                     | 108 |
| Figure 5.6 | '3D' display without hidden surface removal .....  | 109 |
| Figure 5.7 | '3D' display with hidden surface removal .....     | 110 |
| Figure 5.8 | Rotation of '3D' model about x, y and z axis ..... | 111 |
| Figure 5.9 | Zooming in feature of the '3D' model .....         | 112 |

## CHAPTER 1: INTRODUCTION

Magnetic resonance imaging (MRI) generates high resolution, diagnostic quality medical images without ionizing radiation. MRI allows imaging directly in three orthogonal planes: transverse, sagittal, and coronal. A valuable quality inherent in magnetic resonance imaging is its ability to show anatomical structures in gray levels which differentiate between bone and soft tissue as well as between different types of soft tissues. This property is useful both for soft tissue analysis and skeletal studies.

Magnetic resonance imaging is used in a wide variety of applications including such area as chemical analysis, study of structures in solids, fluid flow measurements and biomedical investigations. In the clinical environment, it has become a powerful analytical tool for investigating diseases involving the brain, the spinal cord, the musculoskeletal system, the spine and the cardiovascular system. Craniosynostosis is the premature closure of the sutures of the skull; scaphocephaly refers to the premature closure of the sagittal suture resulting in a head that is both long and narrow[5]. Earlier studies documented raised intracranial pressure and retarded intellect in patients with craniosynostosis[3,5]. Although considerable research has been done on craniosynostosis, the exact cause is still unknown[3,5]. The re-shaping of the skull by surgical treatment has been advocated, but Venes and Sayers[10] indicated that scaphocephaly does not significantly alter brain growth; therefore, the purpose of surgical treatment is mainly cosmetic improvement. In Alberta, Dr. T. Myles[6] studied the effectiveness of the surgical treatment for children with craniosynostosis over the past 15 years. He found that the timing of surgery seems more important in determining the cosmetic result than the type of procedure performed. The planning and evaluation of

these surgical operations are dependent on magnetic resonance imaging for defining the underlying bony structures and their relationship to overlying soft tissue contours. Unfortunately, the planar format of magnetic resonance imaging does not provide adequate information about the three-dimensional structures. Three-dimensional reconstruction software provides a tool to evaluate and visualize the three-dimensional bony architecture of the cranial vault.

There is a large body of literature reporting the development of three-dimensional reconstruction software [1,2,4,6,8,9] for clinical applications. In general, these approaches are complex, time consuming, and require special hardware and software for the reconstruction process. For example, the application developed by Smith *et al.* [9] uses a graphics workstation consisting of a Data General MV/8000 super mini-computer and an ADAGE RD 3000 graphics system. The system developed by Hoffmeister [4] requires a VAX-11/750 with International Imaging Systems (IIS) System 600 software on a IIS Model 75 image processor. These specialized hardware and software resources are not available to most physicians, thus reducing the general accessibility of these three-dimensional reconstruction applications. Manufacturers of magnetic resonance imaging units have also developed their own three-dimensional reconstruction software (for example, 3DCT by Siemens) for their machines. However, it is not economical to run the three-dimensional reconstruction application on an expensive magnetic resonance imaging machine which is heavily used for the medical data collection.

The purpose of this project was to develop a three-dimensional reconstruction application which will run on a desktop computer. This software application provides greater accessibility to clinicians, lowers the operation cost because the radiological imaging and three-dimensional reconstruction processes are separated, and enhances the

possibility of combining computed tomography and magnetic resonance images in one application.

The input data are sets of parallel two-dimensional magnetic resonance images taken along the  $z$  axis with a fixed axial displacement, as shown in Figure 1.1.

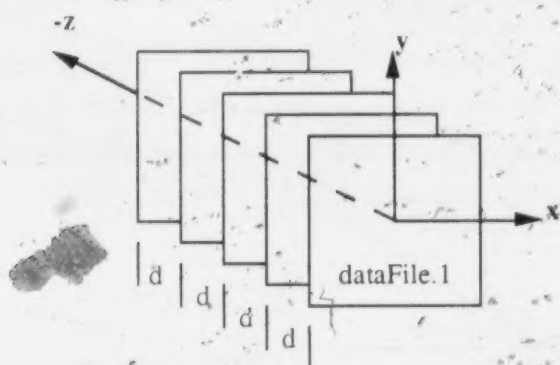


**Figure 1.1:** The input data set is a series of coplanar images with fixed displacement ( $d$ ) along the  $z$  axis.

The  $x$  and  $y$  axes correspond to the horizontal and vertical directions of individual slices. Each image is a collection of samples  $\{ f(x,y) \}$  and is represented as a two-dimensional array of numbers. Each element of this array is called a picture element or pixel which is the amplitude corresponding to the brightness of a small elemental area in the digital image. The image gray scale is defined by digital values; in this project, the gray scale for white is zero and for black is 255.

Following in Chapter 2 is a presentation of the basic principles of magnetic resonance imaging. This includes a description of the imaging techniques and the parameters that affect the image quality. Also, the quantitative errors of the magnetic resonance images produced on the Phillips Gyroscan S15 system used in this project are discussed.

Analysis of image processing techniques for magnetic resonance images of the skull is presented in Chapter 3. The scope of the software application is listed followed by a discussion of spatial-domain operations for data extraction. Then the implementation strategy of a three-dimensional model is presented. The three-dimensional reconstruction application is implemented by two programs: 'HEAD' and '3D'. 'HEAD' performs the boundary extraction function and '3D' creates the three-dimensional model.

Chapter 4 describes the software structure and menu commands. The concept of an event-driven computer is introduced. The software structure and menu commands of 'HEAD' and '3D' are described along with relevant structure charts. The software programs are demonstrated in Chapter 5. The user is guided through the boundary extraction and the three-dimensional reconstruction processes. Results of command menu selections are presented.

Chapter 6 presents the conclusions of this thesis and the direction for further research in the area of three-dimensional display of medical data.

## References

- [1] Biondetti P.R. Vannier M.W., Gilula L.A., & Knapp R.H. "Three-Dimensional Surface Reconstruction of the Carpal Bones from CT Scans", *Comp. Med. Imag. and Graphics*, Vol. 12, No. 1, 1988, pp. 67-73.
- [2] Goldwasser S.M. Reynolds R.A., Talton D.A., & Walsh E.S. "Techniques for the Rapid Display and Manipulation of 3-D Biomedical Data", *Comp. Med. Imag. and Graphics*, Vol. 12, No. 1, 1988, pp. 1-24.
- [3] Hoffman H.J. Hendrick E.B., & Munro I.R., "Craniosynostosis and Craniofacial Surgery", *Oxycephaly*, *Edinburgh Med J.* 33:189, 280, 357, 1982.
- [4] Hoffmeister J.W. Rinehart G.C., Vannier M.W. "Three-Dimensional Surface Reconstructions Using a General Purpose Image Processing System", *Computer Assisted Radiology*, Proceedings of the 3rd International Symposium, 1989, pp.752-757.
- [5] Mohr G. Hoffman H.J., Munro I.R., Hendrick E.B., & Humphreys R.P. "Surgical Management of Unilateral and Bilateral Coronal Craniosynostosis: 21 Years of Experience", *Neurosurgery*, Vol. 2, No. 2, 1978, pp. 83-92.
- [6] Myles S.T. "Craniosynostosis - Review of Surgical Treatment in Southern Alberta", *Seminar in the Development and Treatment of Cranial Deformities*, University of Alberta Hospital, April 26, 1990.
- [7] Richard M.J. Allard J., Ghosh S.K., & Bougouss M. "Three-Dimensional Reconstruction of Human Limbs from Tomographic Views", *Computers and Biomedical Research* 22, 1989, pp 26-35.
- [8] Robb R.A. Barillot C., "Interactive Display and Analysis of 3-D Medical Images", *IEEE Transactions on Medical Imaging*, Vol 8, No.3, Sept. 1989.
- [9] Smith W.K. Schlusberg D.S., Woodward D.J., & Parley R.W. "Use of Computed Tomography for a Three-dimensional Treatment Planning System", *Comp. Med. Imag. and Graphics*, Vol. 12, No. 1, 1988, pp. 25-32.
- [10] Venes J.L. Sayers M.P., "Sagittal Synostectomy", *J. Neurosurg*, Vol. 44, March 1976, pp. 390-393.

## CHAPTER 2: BRIEF REVIEW OF MAGNETIC RESONANCE IMAGING

### 2.1 Background

Magnetic resonance imaging (MRI) relies on the paramagnetic properties of the nucleons ( neutrons and protons ) which make up atomic nuclei. Each neutron and proton possesses an intrinsic spin which produces a magnetic moment causing the nucleus to behave like a tiny bar magnet. When a uniform magnetic field is applied to the nuclei, magnetic coupling occurs which tends to align them with the field direction and causes precession - rotation of the spin axes around the field direction. The rate or frequency of this precession is proportional to the applied magnetic field strength. During the precession state, the nuclei emit a radio-frequency ( RF ) signal, the "Larmor frequency", which has the same frequency as the frequency of precession[4]. The Larmor frequency is related to the applied magnetic field by the Larmor equation:

$$\omega_0 = -\gamma B_0 \quad (2.1)$$

where  $\omega_0$  is the frequency of precession;

$\gamma$  is the gyromagnetic ratio of the magnetic moment of the nuclei  
to its angular momentum; and

$B_0$  is the applied external magnetic field.

The Larmor frequency is constant for nuclei of the same material; that is, all identical nuclei emit a signal of the same frequency in a given magnetic field.

Three types of magnetic fields are required in magnetic resonance imaging. The first field is the static magnetic field,  $B_0$ , which is uniform throughout the imaging sample. The direction of this field is defined to be the Z-axis. The second field is the gradient field which produces a magnetic field profile so that the magnetic resonance imaging signal can be spatially decoded. It consists of a set of three orthogonal field gradients  $G_x$ ,  $G_y$ , and  $G_z$  which generate a spatially-varying and time-varying magnetic field within the sample. For example,  $G_x$  in the X-direction causes the Z-component of the magnetic field to vary linearly with X and the Larmor frequency becomes:

$$\omega = -\gamma (B_0 + xG_x) \quad (2.2)$$

The third field is the radio-frequency field which tips the magnetic moment of the nuclei by an angle  $\theta$  from the Z axis. During the tipping process, the nuclei absorb energy from the radio-frequency field and are shifted to a higher energy state. When the radio-frequency fields stop, the nuclei return to a low energy state by precession and a radio-frequency signal is emitted. This signal provides the image data.

The location and orientation of the image plane is identified by selective excitation in which only those nuclei that possess Larmor frequency equal to the radio-frequency excitation pulse will be affected. Similarly, by applying combinations of three orthogonal gradients and different radio-frequency pulses, spatial information of the image plane is decoded and recorded. After the data is collected, Fourier transformation is applied to convert the data from the time domain to the frequency domain which maps the physical location to a frequency space. The value of each pixel in the image is related to the amplitude of the signal at the location  $x, y, z$  which represents a composite of nuclear densities.

Four tissue properties influence the strength of the imaging signal and the appearance of an image. These properties are the hydrogen concentration, the blood flow rate, the longitudinal relaxation time ( $T_1$ ), and the transverse relaxation time ( $T_2$ ). When the magnetic moment of a nuclei is tilted by a pulsed magnetic field, the nuclei return to the equilibrium state by releasing energy to the lattice of atoms around the nuclei. The time required to return to equilibrium is characterized by  $T_1$  or spin-lattice relaxation time. Because the process occurs in relation to the Z axis (longitudinal orientation),  $T_1$  is also called the longitudinal relaxation time.  $T_2$  or spin-spin relaxation time is the time constant that characterizes the process of returning to equilibrium of the component that is perpendicular to the external static field. Because the process occurs in relation to the XY magnetization which is a plane transverse to external static field,  $T_2$  is also called transverse relaxation time.

Two programmable parameters,  $T_r$  and  $T_e$  also affect the intensity of an image.  $T_r$  is the repetition time, which is the time between two consecutive pulse sequences.  $T_e$  is called the echo time which is the time between consecutive maximum signals in a repeated pulse sequence. In general, the intensity of an image is a function of  $T_1$ ,  $T_2$ ,  $T_r$  and  $T_e$ [2].

By selecting different values of  $T_r$  and  $T_e$ , two types of images can be obtained. A  $T_1$ -weighted image is generated by applying short  $T_r$  and  $T_e$  values.  $T_2$ -weighted images are generated by using relatively long  $T_r$  and  $T_e$  values. In  $T_1$ -weighted images, tissues with short  $T_1$  will appear bright and tissues with long  $T_1$  will appear dark. While on  $T_2$ -weighted images, tissues with long  $T_2$  will appear bright and tissues with short  $T_2$  will appear dark[4].

In summary, magnetic resonance imaging involves the following steps:

- 1) positioning an object in a uniform magnetic field  $B_0$ ;
- 2) selecting planes of interest by applying gradient fields;
- 3) exciting the nuclei in these selected planes by radio-frequency pulses;
- 4) decoding the signal which satisfies the Larmor equation and collecting these data; and
- 5) generating images using Fourier Transform reconstruction techniques.

## 2.2 Scanning Sequence

Several different scanning sequences have been designed to measure specific magnetic resonance imaging parameters such as T1 or T2; the two most commonly used scanning sequences are spin echo and inversion recovery.

### 2.2.1 Spin Echo

Spin echo is the most commonly used pulse sequence in clinical applications[1,3,6]. A 90 degree radio-frequency pulse, which tips the magnetic moment by 90 degree from Z axis, is followed by a 180 degree inverting pulse at time  $T_e/2$  and the spin echo is recorded at time  $T_e$ , as shown in Figure 2.1. The Z gradient is applied during the 90 degree excitation, a signal is emitted at this stage but is ignored; after time  $T_e/2$ , a 180 degree inverting pulse is applied and the Z gradient is re-applied during this process for slice selection. The echo occurs after another  $T_e/2$  time and the image data is recorded. Multiple echoes can be obtained with repeated 180 degree pulses.

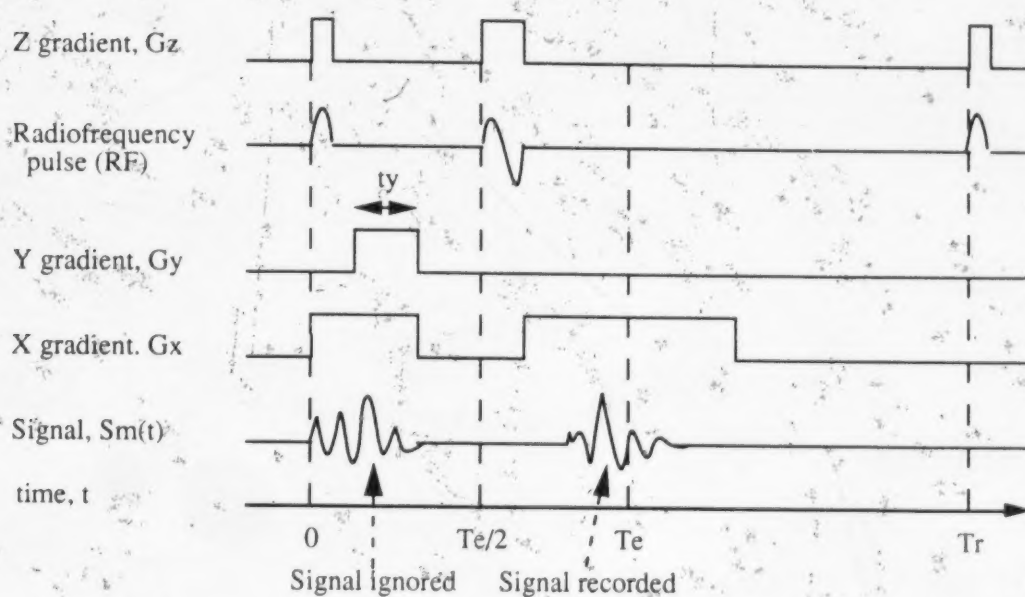


Figure 2.1: The spin echo pulse sequence.

The intensity<sup>[2]</sup> (I) of a spin echo is :

$$I = N(H) * f(v) * (1 - \exp(-Tr / T1)) * \exp(-Te / T2) \quad (2.3)$$

where  $N(H)$  is the hydrogen density ;

$f(v)$  is the flow rate of blood;

$Tr$ ,  $Te$ ,  $T1$  and  $T2$  are as defined in the text.

The intensity of the image is influenced by the tissue being examined; that is, the hydrogen density. For example, hard bone and air spaces (low hydrogen content) which do not produce strong signals are dark while fat and soft tissue (high hydrogen content)

which have high proton densities produce a bright image. The intensity will increase as T2 and hydrogen density increase and as T1 decreases. It should be noted that the T1 and T2 influences are both relative to Tr and Te. The images are weighted by altering the repetition and echo times.

### 2.2.2 Inversion Recovery

Inversion recovery is the other most commonly used imaging pulse sequence. The application of a 180 degree pulse, followed by a 90 degree pulse tips the net magnetization vector from the Z axis onto the X-Y plane. Like spin echo, the Z gradient is applied both during the 180 degree pulse and the 90 degree pulse for slice selection (shown in Figure 2.2). The X and Y gradients are both on during the emission of the magnetic resonance imaging signal to decode spatial location. Varying the value of T1 in inversion recovery can produce a near zero intensity for any specific T1 value which is useful for contrasting adjacent tissues. The signal intensity for inversion recovery is [2]:

$$I = N(H) * (1 - 2 * \exp(-Tr/T1) * \exp(-Te/T2)) \quad (2.4)$$

T2 is assumed to be long with respect to T1.

In summary, the spin-echo sequence is used to obtain T2 dependent images that are not strongly influenced by field inhomogeneity. Images obtained using this sequence have a strong T2 dependence, which is largely determined by the value of Te. The Inversion-Recovery sequence is used to obtain T1 dependent images. This sequence is useful for contrasting adjacent tissue by varying Tr to make the intensity of any tissue with a certain T1 value be zero. Images produced in different magnetic field strengths with different Tr and Te values will have different intensity levels; therefore, magnetic resonance images with different imaging parameters can not be directly compared.

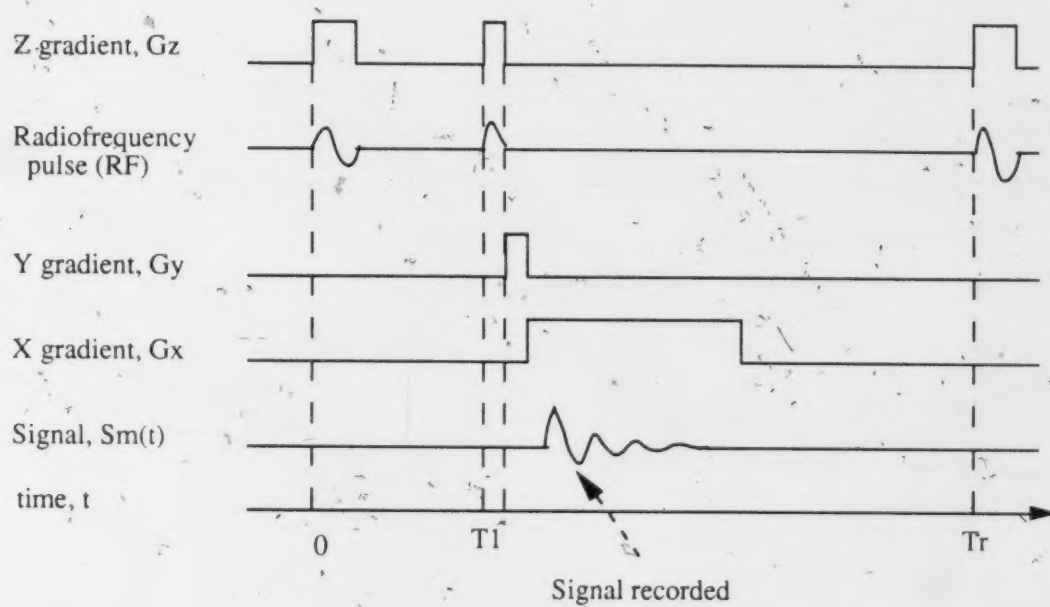


**Figure 2.2:** The inversion recovery pulse sequence.

### 2.3 Spatial Errors of Magnetic Resonance Images

A thorough error analysis of the data acquisition phase and suggestions for correcting system dependent errors is presented by Unrau[5]. The emphasis here is on the overall spatial accuracy of the image produced. An error analysis has been carried out using a plate of grid lines spaced at 1 cm intervals. The images are taken one year apart to verify image repeatability using a Phillips Gyroscan S15 1.5 Tesla magnetic resonance imaging scanner. The image obtained consists of the grid lines and noise around the edges of the plate and in the background, as shown in Figure 2.3.

The image field of view is 250 millimetres and the image resolution is 256 pixels; that is, 256 pixels represents 250 millimetres. By measuring the number of intervals and the number of pixels between the end points of each row and column along each grid line, the number of pixels per 250 millimetres were calculated (Appendix A (1)). Then the percentage error of each row and column was calculated with respect to the field of view and the image resolution. The percentage errors of the phantom measurements taken at two separate times using the same imaging parameters are shown in Table 2.1.

From the percentage columns in Table 2.1, there is minimal error in the middle section of the phantom. Since the grid lines may not aligned with the coordinate axes of the digital image, the number of pixels for each gridline has an accuracy of -1 to +1 pixel; that is for each grid line, a maximum measurement error of one pixel per grid line may be introduced; therefore,

$$\begin{aligned} \text{grid line error in percentage is} &= (1 / 256) * 100\% \\ &= 0.4\% \end{aligned}$$

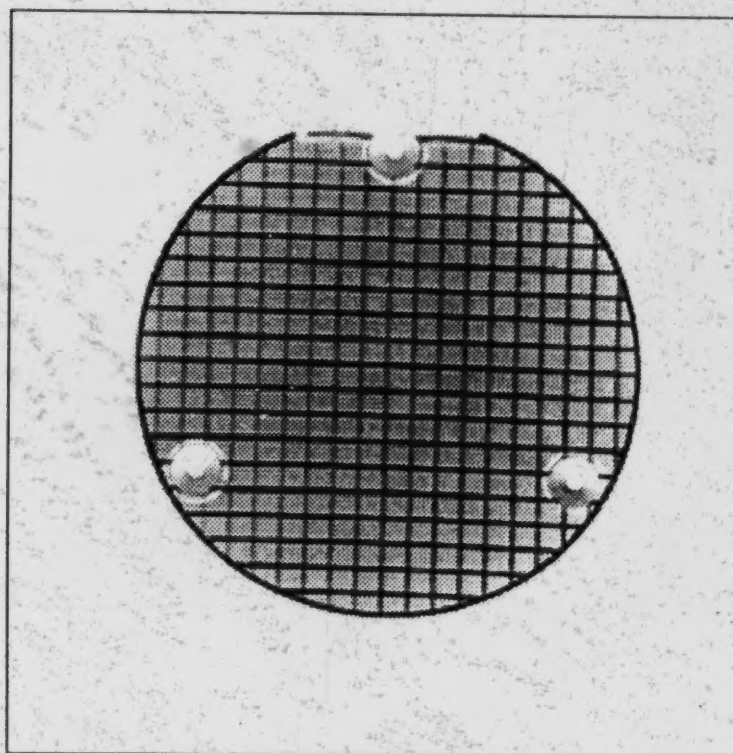


**Figure 2.3:** Test image for error measurement.

Table 2.1: Percentage Errors of Gridline Measurements.

| Gridline  | Phantom Image 1               |                                 | Phantom Image 2               |                                 |
|-----------|-------------------------------|---------------------------------|-------------------------------|---------------------------------|
|           | Vertical<br>Gridline<br>( % ) | Horizontal<br>Gridline<br>( % ) | Vertical<br>Gridline<br>( % ) | Horizontal<br>Gridline<br>( % ) |
| 1         | -0.8                          | -0.4                            | -0.8                          | -0.4                            |
| 2         | 0.0                           | 0.0                             | 0.0                           | 0.0                             |
| 3         | -0.8                          | -0.4                            | -0.8                          | -0.4                            |
| 4         | -0.8                          | 0.0                             | -0.8                          | 0.0                             |
| 5         | -0.4                          | 0.0                             | -0.4                          | 0.0                             |
| 6         | -0.4                          | 0.0                             | -0.4                          | 0.0                             |
| 7         | -0.8                          | 0.0                             | -0.8                          | 0.0                             |
| 8         | -0.8                          | 0.4                             | -0.8                          | 0.4                             |
| 9         | -0.4                          | 0.4                             | -0.8                          | 0.4                             |
| 10        | -0.4                          | 0.4                             | -0.8                          | 0.0                             |
| 11        | 0.0                           | 0.4                             | 0.0                           | 0.0                             |
| 12        | 0.0                           | 0.4                             | 0.0                           | 0.0                             |
| 13        | 0.0                           | 0.4                             | 0.0                           | 0.0                             |
| 14        | -0.8                          | 0.4                             | 0.0                           | 0.0                             |
| 15        | -0.4                          | -0.4                            | -0.4                          | -0.4                            |
| 16        | -0.4                          | -0.4                            | -0.4                          | -0.4                            |
| 17        | -0.8                          | 0.0                             | 0.0                           | 0.0                             |
| 18        | -0.8                          | 0.8                             | -0.8                          | 0.8                             |
| 19        | -0.4                          | 0.8                             | -0.4                          | 0.8                             |
| 20        | -0.8                          | ----                            | -0.8                          | ----                            |
| mean      | -0.50                         | 0.15                            | -0.46                         | 0.04                            |
| Std. Dev. | 0.31                          | 0.38                            | 0.35                          | 0.35                            |

Assuming that all subsequent processing steps are ideal, the maximum error in the X and Y direction is equal to 1.2%, grid line error ( 0.4% ) plus maximum measurement error ( 0.8% ). The static magnetic field  $B_0$  in the imaging volume is extremely accurate which in turn produces a very accurate measurement in the Z direction; the error is immeasurably small[5].

The Student's t-test was used to test for significant differences between the two horizontal gridlines measurements and the two vertical gridlines measurements over time. The null hypothesis was that there is no significant differences between the gridlines measurements of two phantoms over time. For the vertical gridlines measurements, the computed t-value (  $t_c$  ) is 0.623. For a level of significance ( $\alpha$ ) of 0.05, the critical values for a two-sided alternative are  $t = 2.101$ (1). Since  $t_c$  falls between these values, the null hypothesis that there is no significant difference between the two vertical gridlines measurements is accepted. For the horizontal gridlines, the computed t-value (  $t_c$  ) is 1.837. For a level of significance ( $\alpha$ ) of 0.05, the critical values for a two-sided alternative are  $t = 2.093$ (1). Since  $t_c$  falls between these values, the null hypothesis that there is no significant difference between the two horizontal gridlines measurements is accepted. From the results of t-test, the differences in the measurements are not large enough to reject the assumption that the measured data are equal at  $\alpha=0.05$  level of significance. Therefore, the spatial errors do not vary over time.

---

(1) From t-distribution table in Statistical Method, Donald L. Harnett, Addison-Wesley Publishing Company, Inc. 1982, p. A-49.

## 2.4 Magnetic Resonance Images of Scaphocephalic head

Figure 2.4 and 2.5 are two samples from a set of T2-weighted magnetic resonance images of an infant's head with scaphocephaly. Part (a) is a gray scale display of the anatomical structure of the skull. This gray scale is directly proportional to the hydrogen density in the tissue; therefore, structures having a high, intermediate, or very low hydrogen density appear white, gray, or black, respectively. The gray portion in the middle of the image is the soft tissue of the brain and the black boundary is the bone of the skull. Parts (b) and (c) are the gray-level profile plots of the image (shown in part (a)) along AA' and BB'.

Ideally, the bone would have a constant gray level in the image; but Figure 2.4(b), (c) and Figure 2.5(b), (c) show that the gray level of the bone in fact varies from 255 to 110. For most cases, there is a distinct gray level difference between the bone and the surrounding structures. But in some parts of the images ( shown in Figure 2.5(c) ), the bone gray level is very close to that of the surrounding structures which makes it difficult to separate the bone anatomy from the soft tissue anatomy.

Gaps exist in the skull due to the opening of sutures as shown in Figure 2.5(a). A discontinuity of the skull is a special case in boundary extraction because the extraction process has to determine whether it is a true gap, such as a suture opening of the skull, or low image quality. Also, there are low gray-level variations in the background which will affect the boundary extraction process. Approaches to overcome these problems are discussed in Chapter 3.

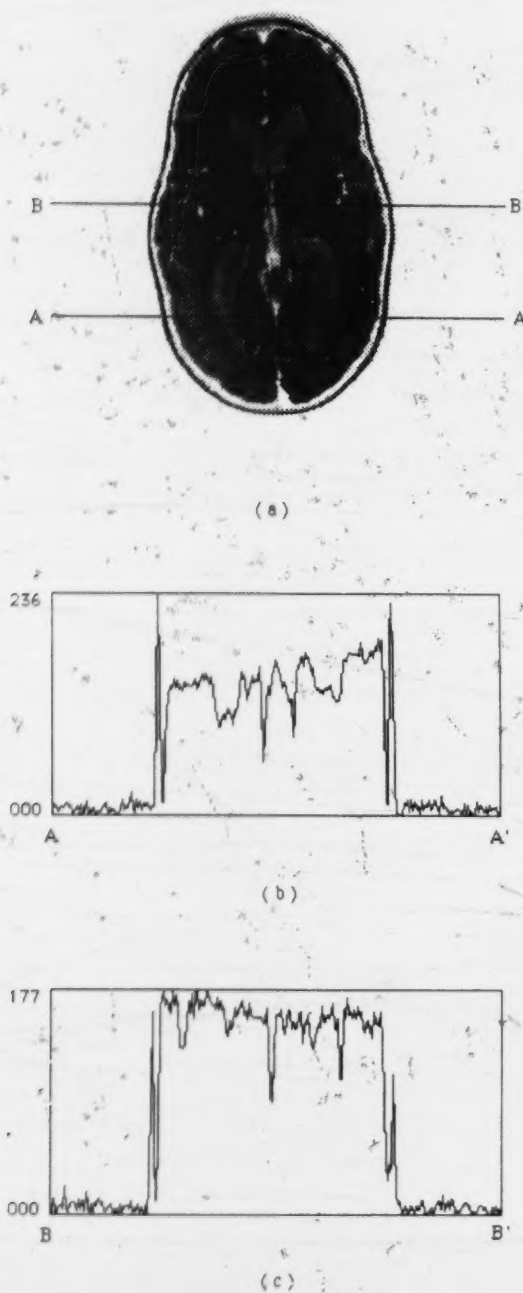


**Figure 2.4:** (a) Magnetic resonance image.  
(b) Gray-level profile plot along AA' and BB'.

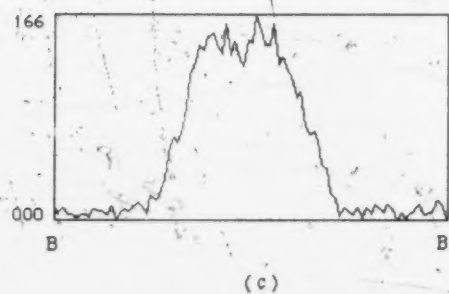
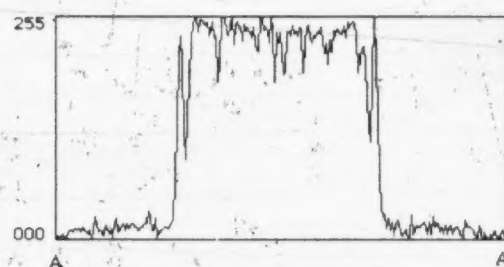
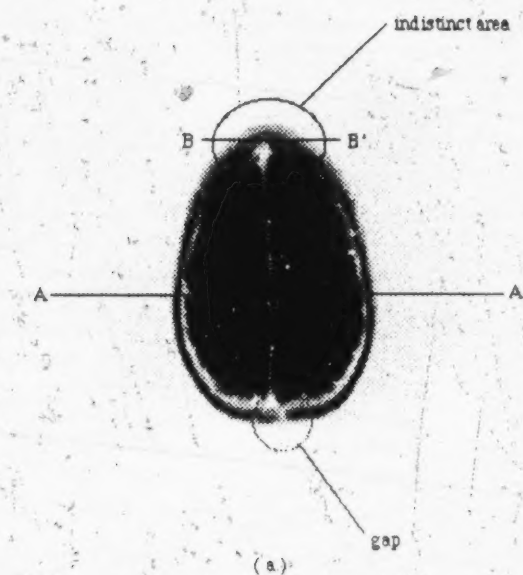


Figure 2.5: (a) Magnetic resonance image.  
(b) Gray-level profile plot along AA' and BB'.

## References

- [1] Bushong S.C., Magnetic Resonance Imaging, The S.V. Mosby Company, St. Louis, 1988.
- [2] Crooks L.E., "Image Contrast Mechanisms in MRI", Medical Magnetic Resonance imaging and Spectroscopy, 1986, pp. 36-47.
- [3] Maudsley A.A., "Principles of Nuclear Magnetic Resonance", Medical Magnetic Resonance imaging and Spectroscopy, 1986, pp. 3-13.
- [4] Morris P.G., "Nuclear Magnetic Resonance Imaging in Medicine and Biology", 1986, Oxford University Press, New York.
- [5] Unrau R.C., "Quantitative Analysis of Nuclear Magnetic Resonance Imaging", Master Thesis of University of Toronto, 1988.
- [6] Young I.R., "Imaging Techniques and Reconstruction Methods", Medical Magnetic Resonance imaging and Spectroscopy, 1986, pp. 14-35.

## CHAPTER 3: IMAGE PROCESSING TECHNIQUES FOR MAGNETIC RESONANCE IMAGES OF THE SKULL

### 3.1 Scope of the software application

The objective of this project was to develop a software application which will extract boundary information from magnetic resonance images and produce a three-dimensional reconstructed wire-frame display. The scope was to:

1. automate the boundary extraction process for a set of magnetic resonance images of scaphocephalic infant skull images stored in Tag Image File Format<sup>[1]</sup>;
2. provide image processing capability during this boundary extraction process;
3. reconstruct a three-dimensional wire-frame model from the boundary data;
4. display the three-dimensional model with or without hidden surface removal;
5. provide controls for rotating the three-dimensional model about x, y, and z axis in real time;
6. provide a user-friendly interface to ease the use of the application; and
7. run the application in a desktop environment.

### 3.2 Development environment

The block diagram of the software is shown in Figure 3.1. The input data is a set of magnetic resonance images (image data set) converted into Tag Image File Format. The boundary data extracted from these data sets are used in the three dimensional reconstruction process. The boundary extraction process is applied only once and the extracted data is then stored for later processing as needed. Two separate programs were

developed; 'HEAD' for the boundary extraction process, and '3D' for the three dimensional reconstruction process. 'HEAD' reads the image data set as input data, provides filtering operations to enhance the image, extracts the boundary data, and stores the boundary data in a file. The boundary data produced by 'HEAD' is used by '3D' to perform three dimensional reconstruction which is then displayed with or without hidden surface removal.

User-friendliness and ease of use were achieved by using the graphical user interface, the mouse, and the command-key of the Macintosh computer. Controls for three dimension rotation are provided by control scrollbars in the display window. The minimum memory requirement was achieved by dynamically allocating memory space using pointers for the data structure in both programs.

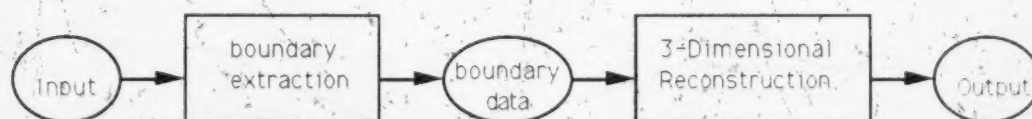


Figure 3.1: The flow diagram of three-dimensional reconstruction.

Both application programs were developed on a Macintosh<sup>1</sup> II computer with 8 megabytes of main memory, an 80 megabyte internal hard drive, and running under MultiFinder<sup>1</sup>. The development tool was Macintosh Programmer's Workshop<sup>1</sup> (MPW).

v. 3.0 and the debugging tool was Symbolic Application Debugging Environment<sup>1</sup> (SADE) v. 1.0. 'HEAD' is written in MPW Pascal and '3D' is written in MPW C. Both applications require at least 2 megabytes of memory, but 4 megabytes or more is recommended when using MultiFinder.

### 3.3 Input File Requirements

The magnetic resonance imaging data are stored in straight binary format with an image size of 256 pixels x 256 pixels x 16 bits per pixel. The header file for each slice is discarded and the parameters of the images taken from a parameter file are entered manually. The parameters used by this application are slice thickness, pixel size, number of pixels, and inter-slice factor. Each pixel represents the average value of the intensity of the imaging volume at that particular location and the slice thickness is the thickness of this imaging volume for each image. The inter-slice factor specifies the distance between the centers of two consecutive imaging volumes in terms of slice thickness. That is, for a slice thickness of 10 mm and an inter-slice factor of 1.2, the distance between the centre of two consecutive slices is 12 mm ( $10\text{mm} * 1.2$ ). Pre-processing is required to convert the pixel size from 16 bits to 8 bits per pixel and to store the data in Tag Image File Format<sup>[1]</sup> (TIFF). An image data set consists of between 12 to 15 images. The image files are sequenced in ascending order or descending order having the same file name plus the suffix of .n, where n ranges from 1 to the number of slices. Slices in a set of magnetic resonance image files are taken along the same z-axis and it is assumed that there is no misalignment in the image-to-image registration.

---

<sup>1</sup> Macintosh, MultiFinder, Macintosh Programmer's Workshop and Symbolic Application Debugging Environment are trademarks of Apple Computer, Inc.

### 3.4 Skull Boundary Extraction

The first step in three-dimensional reconstruction is to identify and extract the boundaries of regions of interest. A number of algorithms have been proposed in the literature [2,6,7,11,16,17]. All algorithms consist of two stages: an segmentation step followed by a contour extraction step.

#### 3.4.1 Image Segmentation

The segmentation process decomposes an image into regions that can be identified. A region is defined as a set of spatially connected elements that have certain common characteristics that differ from the characteristics of adjacent regions. Based on the similarity or discontinuity between regions, segmentation techniques are categorized into three classes<sup>[10]</sup>: 1) thresholding; 2) edge detection; and 3) region growing.

Thresholding is the most basic method of extracting objects from an image. The result of thresholding is defined as<sup>[20]</sup>:

$$\begin{aligned} g(x, y) &= 1 && \text{if } f(x, y) \geq t; \\ &= 0 && \text{if } f(x, y) < t; \end{aligned} \quad (3.1)$$

where  $(x, y)$  are the  $x$  and  $y$  coordinates of a pixel;  $g(x, y)$  and  $f(x, y)$  are the segmented and the gray level functions of  $(x, y)$  respectively;  $t$  is the threshold value. Thresholding works well in situations where the gray level of a region is uniform or where there is a distinct difference in gray levels between neighbouring regions. Most images have a distinct gray level difference between the skull and the surroundings, and a suitable threshold value can be easily chosen. In the indistinct parts of the images (Figure

2.5(b)), the difference in gray level is small and selecting a suitable threshold value to separate different structures presents difficulties.

The gray level of bone varies from 255 to 110 in Figure 3.2(a). To detect the skull, the minimum threshold value of 110 is used. This threshold value eliminates the background noise, extracts the bone, and highlights the soft tissue with pixel values above 110 ( Figure 3.2(b) ). For this image, there is no loss of boundary information because the bone is surrounded by structures which have gray level below 110 and can be separated by thresholding. However, when the same threshold value is applied to the image in Figure 3.3(a) ( same as Figure 2.5(a) ), thresholding does not separate the bone from surrounding structures (Figure 3.3(b)) because the gray levels of bone are very close to the gray level of the surrounding structures in the indistinct part of the skull. Consequently, both bone and soft tissue merged together into a single region. If a higher threshold value is used ( 150 ), the soft tissue as well as the bone in the indistinct part of the skull are lost ( Figure 3.3(c) ). Therefore, thresholding is not sufficient in detecting boundaries for the images used in this project because it does not lend itself to automatic processing of all the images in an image data set. But thresholding is useful in removing low gray level variation found in the background, because the structures of the skull in this image data set have a gray level above 80 while the background has gray level below 80. By setting the threshold value to 80, the background gray level variation can be eliminated.

Another approach to image segmentation is to divide the image into regions. Region growing uses image characteristics to join individual pixels into larger regions. Region growing techniques can be broken down into: 1) local techniques, 2) global techniques, and 3) splitting and merging techniques.[3]



(a)



(b)

**Figure 3.2:** (a) Magnetic resonance image.  
(b) Result of applying threshold value of 110.



**Figure 3.3:** (a) Magnetic resonance image.  
(b) Result of applying threshold value of 110.  
(c) Result of applying threshold value of 150.

Local techniques join pixels into regions based on their pixel gray level or the gray level of their neighbours. Global techniques group pixels into regions based on the gray level of pixels distributed throughout the image. Splitting and merging techniques first divide the whole image into unit regions of equal size. Then the unit region is split into sub-regions if the homogeneity property is not met in the unit region, or merge into a larger region if the homogeneity property is met. Gray level is often used as the homogeneity criterion. A common problem with region growing techniques occurs when the image has a background of varying gray levels<sup>[3]</sup>. When applied to our images, such techniques are not able to distinguish bone from surrounding structures at the indistinct part of the image in Figure 2.5(a). Also, region growing techniques are iterative, time consuming, and the regions produced depend on the order in which sub-regions are merged<sup>[10]</sup>.

The edge detection technique is a segmentation technique which uses spatial information to find edges. It is based on the detection of a discontinuity in gray levels between regions. An edge or boundary is considered as the pixel locations of more or less abrupt change in gray level. Edge detection techniques can be divided into two kinds: local and regional. Regional edge detection techniques apply windows several times the width of the edges to be detected. They work well when the image contains large regions of constant gray level and fails if closely adjacent edges are within the window size<sup>[24]</sup>. For example, when several edges appear within the window, regional edge detection techniques do not detect the correct number of edges. Therefore, this technique is not suitable in detecting skull boundaries with closely adjacent edges.

Local edge detection techniques apply small local operators (usually  $3 \times 3$ ) to detect local gray level variations at each point of the image. By using local spatial information to detect edges, local edge detection techniques can detect the gray level variation in indistinct areas and are useful in extracting skull boundaries.

For a continuous image  $f(x,y)$ , its first-order derivative produces a local maximum in the direction of the edge. One edge detection technique measures the slope of  $f(x,y)$  along  $r$  in a direction  $\theta$  and finds the maximum slope of the edge [19]. The maximum slope of  $f(x,y)$  is the gradient of the edge which is perpendicular to the direction of the edge at location  $(x,y)$ . Its mathematical representation is:

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x} \frac{\partial x}{\partial r} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial r} = f_x \cos \theta + f_y \sin \theta \quad (3.2)$$

The maximum slope of  $\partial f / \partial r$  is obtained by setting:

$$\frac{\partial}{\partial \theta} \left( \frac{\partial f}{\partial r} \right) = 0 = -f_x \sin(\theta_g) + f_y \cos(\theta_g) \quad (3.3)$$

By solving equation 3.3:

$$\theta_g = \text{atan} \left( \frac{f_y}{f_x} \right) \quad (3.4)$$

Substituting  $\theta_g$  into Equation 3.2 gives:

$$\left( \frac{\partial f}{\partial r} \right)_{\max} = f_x \cos(\theta_g) + f_y \sin(\theta_g) \quad (3.5)$$

since

$$\cos(\theta_g) = \frac{1}{\sqrt{1 + \left(\tan(\theta_g)\right)^2}} = \frac{\frac{\partial f}{\partial x}}{\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}} \quad (3.6)$$

and

$$\sin(\theta_g) = \frac{\frac{\partial f}{\partial y}}{\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}} \quad (3.7)$$

Substituting  $\cos(\theta_g)$  and  $\sin(\theta_g)$  into Equation 3.5 gives:

$$\left(\frac{\partial f}{\partial r}\right)_{\max} = \sqrt{f_x^2 + f_y^2} \quad (3.8)$$

where  $\theta_g$  is the direction of the maximum rate of change of  $f(x,y)$ .

Based on this concept, gradient operators measure the gradients  $[g_x(x,y)$  and  $g_y(x,y)]$  of the image  $f(x,y)$  in  $x$  and  $y$  directions by a pair of masks  $H_1$  and  $H_2$  respectively. A digital approximation to the gradient of  $f(x,y)$  is calculated by adding the absolute value of these two gradients:

$$g(x,y) = |g_x(x,y)| + |g_y(x,y)| \quad (3.9)$$

Two well known gradient operators are listed in Table 3.1. Both operators compute horizontal and vertical differences of local sums and yield zero for uniform regions.[14]

Table 3.1: Two of the common gradient operators.

| Operator    | Masks                             |                                   |
|-------------|-----------------------------------|-----------------------------------|
|             | H <sub>1</sub><br>( x direction ) | i <sub>2</sub><br>( y direction ) |
| Prewitt[23] | -1 0 1                            | -1 -1 -1                          |
|             | -1 0 1                            | 0 0 0                             |
|             | -1 0 1                            | 1 1 1                             |
| Sobel[8]    | -1 0 1                            | -1 -2 -1                          |
|             | -2 0 2                            | 0 0 0                             |
|             | -1 0 1                            | 1 2 1                             |

Thresholding techniques may then be implemented with an edge detection algorithm to produce a binary image. If  $|g(x,y)|$  is greater than a threshold value  $t$ , the pixel value  $p(x,y)$  is set to black; otherwise, the pixel value is set to white. That is:

$$\begin{aligned}
 p(x,y) &= 255, \text{ if } |g(x,y)| > t; \\
 &= 0, \text{ otherwise;}
 \end{aligned}
 \tag{3.10}$$

The resultant binary image becomes an edge map that contains the necessary data for tracing the object boundary in the image.

A second edge detection technique applies a second-order derivative to detect gray-level transitions. The most common is the Laplacian operator defined as [14]:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (3.11)$$

The Laplacian operator highlights all edges regardless of their orientation but is more sensitive than gradient operators to noise in the background because of the second derivative [14,12]. A digital approximation [19] to the Laplacian of  $f(x,y)$  is:

$$f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y) \quad (3.12)$$

The above approximation considers only the four nearest neighbours while other variations consider the eight nearest neighbours as well. The weighting function in Table 3.2(a) is derived from Equation 3.12 while the other is a variation of a digital approximation derived from Equation 3.11 considering eight nearest neighbours.

**Table 3.2:** Two different weighting functions of Laplacian operator.

| (a)    | (b)[21]  |
|--------|----------|
| 0 1 0  | -1 -1 -1 |
| 1 -4 1 | -1 9 -1  |
| 0 1 0  | -1 -1 -1 |

### 3.4.2 Performance Measurements of Edge Detection Operators

Two approaches were used to examine the performance of edge detection operators. The first measurement computes the ratio ( $r$ ) of the number of original edge points of the input image to the number of incorrect edge points of the output image[24]. The second measurement,  $R$ , computes the weighted and normalized deviation of an actual edge point in the output image from the ideal edge in the input image; that is, how good an detected edge point is.  $R$  is defined by Pratt[22]:

$$R = \frac{1}{I_M} \sum_{i=1}^{I_E} \frac{1}{1 + \delta e_i^2} \quad (3.13)$$

where  $I_E$  = the actual number of detected edge points.

$I_P$  = the number of edge points on the ideal edge.

$I_M = \max \{ I_E, I_P \}$

$\delta$  = scaling constant (defined by Pratt as 1/9).

$e_i$  = distance between an edge point and the ideal edge.

An image of an oval ring which resembles the general shape of a skull was constructed in a background containing low gray-level variation taken from magnetic resonance images. The first set of images had ideal edges ( Figure 3.4(a) ) with line widths of 1, 2, 3, 4, 6, 8, and 10 pixels. The second set of images had ramp edges ( Figure 3.4(b) ) with line widths of 6, 8, 10 and 12 pixels.

### 3.4.3 Results of performance measurements

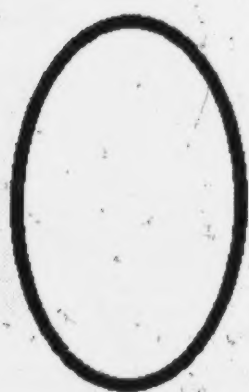
#### a) Ideal edge

Results for the ideal edge are presented in Figure 3.5 and 3.7. Even for an ideal edge, all tested edge detection operators do not find all the edge points. For 1-pixel to 3-pixel line widths, the  $r$  values for all operators are similar, ranging from 6.5 to 7.5. As the line width increases, the  $r$  values increase to the range from 8.2 to 10 but the Prewitt operator generally has a better  $r$  value than the others. The  $R$  values for all operators fluctuate between 0.88 and 0.95 for line widths less than 5-pixels, and become more stable as the line width increases. Although the Sobel operator has the best  $R$  values at 1-pixel, 2-pixel, and 5-pixel widths, its performance varies with line width and appears unstable. In general, the Prewitt operator is more stable and has high  $R$  and  $r$  values.

#### 2. Ramp edge

The results for the ramp edge are shown in Figures 3.6 and 3.8. In general, all operators perform poorly in detecting edges and produce wider edges than in the ideal case. All operators have a stable  $r$  value except the Laplacian (b) operator whose  $r$  value increases from 5.3 (6-pixel line width) to 7.2 (10-pixel line width). From the figures, gradient operators are less sensitive to noise than Laplacian operators and have a higher  $r$  and  $R$  values. The Prewitt operator is more stable and has higher  $r$  and  $R$  values than Sobel operator.

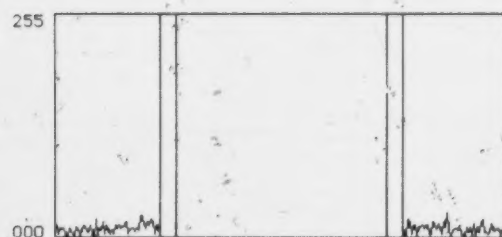
In conclusion, the Prewitt operator appears to be the best of the four operators tested in terms of finding more edge points with high ratio of the number of original edge points to the incorrect detected edge points ( $r$ ) and high weighted and normalized deviation of an actual edge point in the output image from the ideal edge in the input image ( $R$ ).



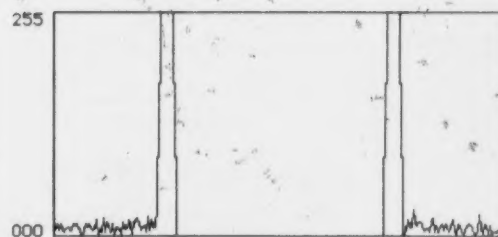
(a)



(b)



(c)



(d)

**Figure 3.4:** Test images of 8 pixel line width. (a) Ideal edge; (b) ramp edge; (c) and (d) corresponding gray-level profile plot.

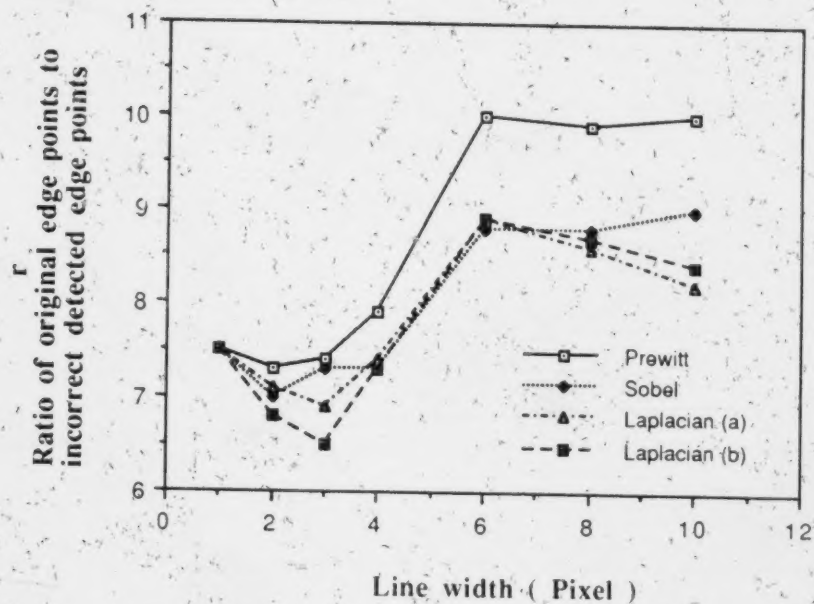


Figure 3.5: Performance measurements of  $r$  for ideal edges.

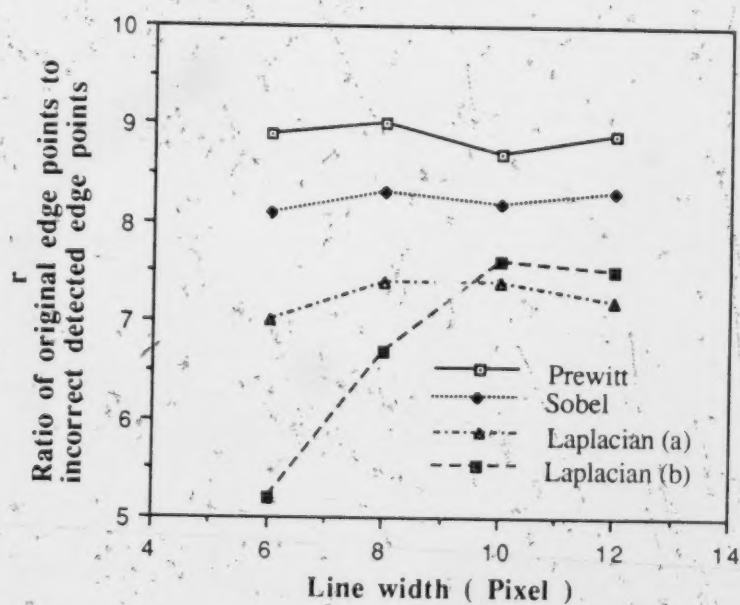


Figure 3.6: Performance measurements of  $r$  for ramp edges.

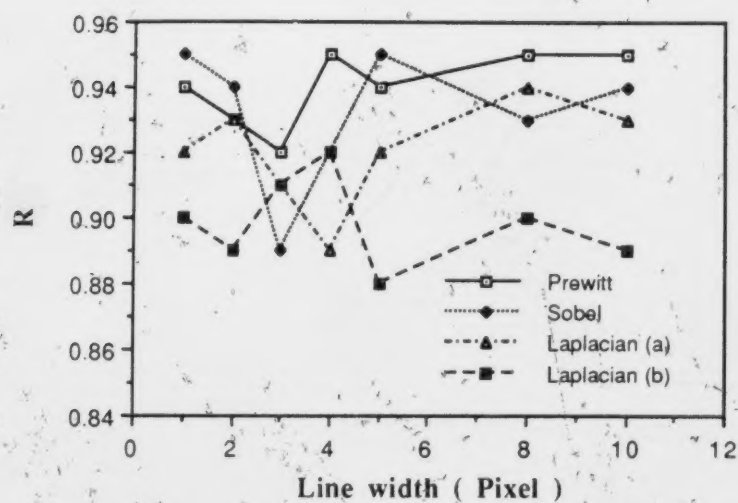


Figure 3.7: Performance measurements of R for ideal edges.

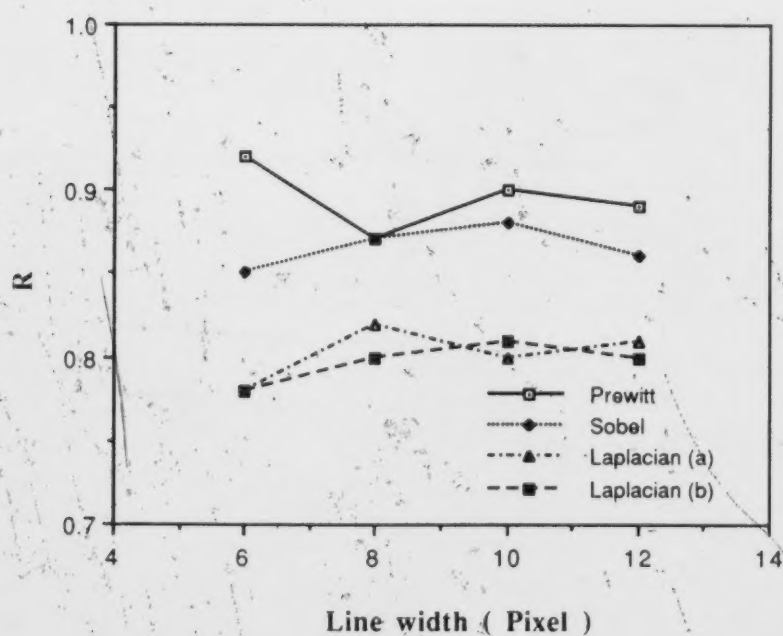


Figure 3.8: Performance measurements of R for ramp edges.

### 3.4.4 Contour Extraction

After obtaining a binary map of the object by segmentation, the next stage is to extract the boundary information of the object. An intuitive approach is to follow the contour and extract shape information of the object. The boundary extraction algorithms used by Keller[16], Udupa[25], and Batintzky[4,5] assume that the contour is a closed loop; therefore, by following the contour, the outer boundary is extracted. In our images, the contour of the skull is not always a closed loop because of open sutures. When the above contour tracing algorithms are used, not only the outer contour but also the inner contour will be extracted because the algorithms follow the edge points at the opening and continues around to the inner boundary. To overcome the inability of the contour tracing algorithm to distinguish between the inner and outer contours of skull, the binary map of skull is thinned to one-pixel-width before the tracing algorithm is applied to the extract boundary information. Assuming that edge points of the outer contour are the outermost points of the object, the thinning process scans each row for the leftmost and rightmost black pixels, and each column for the uppermost and lowermost black pixels. Then the eight neighbours of these outermost black pixels are examined for contour elements. If there are at least three consecutive black neighbours in an clockwise direction, the black pixel is recognized as valid outermost contour point[16]; otherwise, the scanning continues to search for a valid outermost pixel point or until the end of row or column is reached. After finding the outermost pixel points, the thinning process retains the outermost pixel points and eliminates all other pixel points on the binary map. After the thinning process, the contour tracing algorithm is applied to extract boundary information.

A contour tracing algorithm extracts the boundary of an object in the contour map. The process is divided into two steps: starting point selection and next contour

element search [16, 18]. The starting point selection is used to find a seed candidate on the boundary region of interest. If the seed candidate is a valid boundary pixel, the next contour element search will start from the seed candidate. For an object of interest located in the centre of the magnetic resonance image, a search starting at the mid-point of the left-hand border towards the image centre will find the boundary pixel of the object. The first black pixel encountered during the search becomes a seed candidate. Then the eight neighbours of the seed candidate are examined for contour elements. If the upper and lower neighbours contain a contour element, the seed candidate is selected as the starting point of the contour; otherwise, the seed candidate is invalid and assumed to be background noise. Then the search restarts at the right hand side pixel location of the invalid seed candidate until a starting point is found.

The second step searches for the next contour element. After thinning process, the binary map contains contour information only. By scanning each row, the contour elements are extracted. However, this method is not efficient because the number of search for each contour element depends on the location of the contour element in the binary map and generally the number of search is more than 10 pixel locations. Using a 3x3 tracing window is more efficient because the maximum number of search per each contour element is 8 pixel locations and the average number of search is 4 pixel locations [13, 18]. Fig 3.9 shows the tracing window with a centre point  $X_{ij}$  and its eight neighbours, where  $i$  and  $j$  are the  $x$  and  $y$  coordinates of the input image. The eight pixels surrounding the first border element are examined in a counter-clockwise direction. The first black pixel encountered is marked as the next contour element. The tracing window shifts to this new contour element location and the same search sequence is applied to find the next contour element for this new location.

The contour tracing may proceed upwards, downwards, left or right. Contour edges are directional, that is the next boundary pixel will likely be found in the same

|   |   |   |
|---|---|---|
| 8 | 7 | 6 |
| 1 | X | 5 |
| 2 | 3 | 4 |

Figure 3.9: Contour tracing window.

direction. By using this directional feature, the next boundary pixel may be located more quickly by searching from the pixel location next to the previous boundary pixel in the tracing window. For example, if the current contour element is at (row 4, column 5) in Figure 3.10(a) and the next contour element is found in position 5 (row 4, column 6), the search will start from position 2 (row 5, column 5) in a counter-clockwise direction after moving the tracing window to the position of the new contour element (row 4, column 6) as shown in Figure 3.10(b). Position 1 (row 4, column 5) does not need to be searched because it is the previous contour element. In this particular example, position 2 (row 5, column 5) and position 3 (row 5, column 6) in Figure 3.10(b) also need not to be searched because they are already searched in the previous operation. But the starting search location depends not only on the relative position between the previous contour element and the current contour element, but also on the search direction. There is no generalized rule in determining the search start location; therefore, the search algorithm uses a fixed scheme in determining the start location for the simplicity in implementation.

The tracing process starts from the point selected using the search sequence of 2, 3, 4, 5, 6, 7, 8. After the next contour element has been selected, the tracing window moves to the new contour element and uses the prior information of the relative position

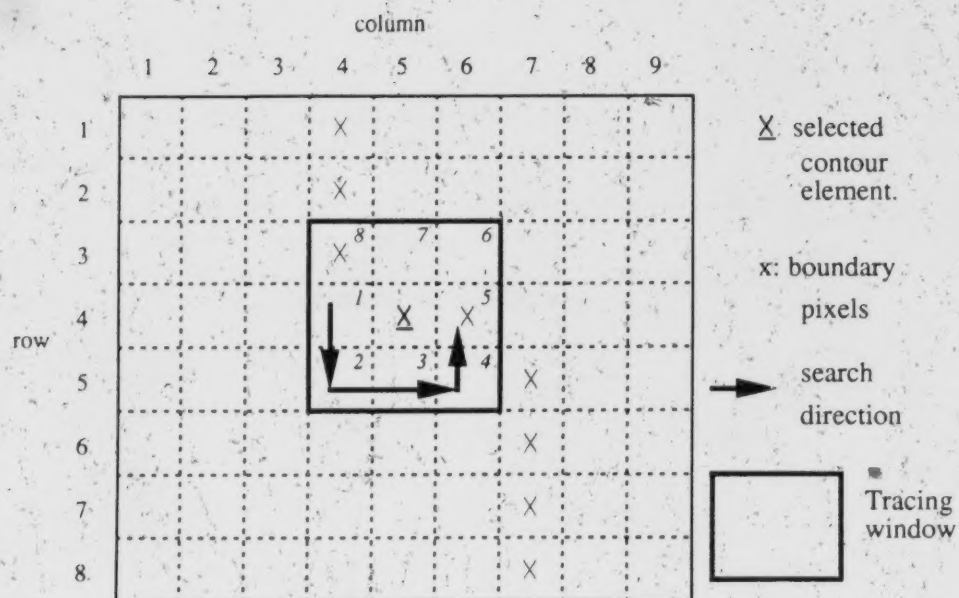


**Figure 3.10(a):** Next contour element is found in position 5 (row 4, column 6).

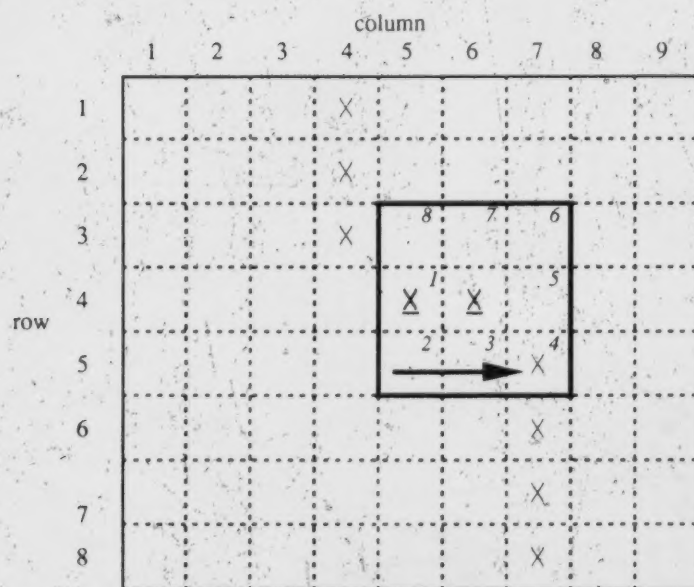


**Figure 3.10(b):** Next contour element search start from position 2 (row 4, column 5).

between the previous and current contour element in determining the search sequence. Boundary searching continues until one of the following termination conditions occurs:

- 1) the next boundary pixel is the same as the starting point, or
- 2) the next boundary pixel is the previous boundary pixel.

Condition 1 implies successful boundary detection with a closed loop. Condition 2 implies a dead end of the boundary detection.

When condition 2 occurs, the tracing algorithm will scan the neighbourhood of the last contour element for the next line segment. Using the directional characteristic of edges, the contour element of the next line segment will likely be found in the direction of the last edge; therefore, the scan area is a semi-circle centred at the last contour element in the direction of the last edge. The radius of the scan area is set to 10 pixels because the width of the gaps in the images are in the range of 1 to 8 pixels. If the next line segment is found, the search will resume at the new line segment; otherwise, the search will terminate and the traced contour elements are displayed.

### 3.5 Skull Boundary Representation

Storing the boundary data in a file allows the same set of data to be used repeatedly without going through the time consuming process of boundary extraction again. Also, if the data file is saved on a floppy disk, the data can be transported easily. A number of encoding methods for the contour are suggested in [7,9,11,18]. All methods require extra processing time for encoding and decoding the data and this is not desirable for desktop computers. For example, code chaining requires the decoding of the code into x, y, and z coordinates. A straight forward method for coding the contours uses X and Y coordinates of the contour elements. Generally, the number of contour elements ranged from 400 to 600; for a set of 15 magnetic resonance images, the maximum

memory requirement is 36,000 bytes ( 15 images \* 600 contour elements/images \* 2 coordinates/contour elements \* 2 bytes/coordinate ) which can be stored on a floppy disk. This method also allows the reading back of the boundary coordinates without any extra processing time. Therefore, the direct coding of contour elements is used for fast processing.

### **3.6 Image Enhancement-by Filtering**

Image enhancement refers to smoothing, sharpening, or modification of images so that features such as edges, lines or contrast become more useful for human visualization or for machine analysis. The enhancement process does not add any additional information to the original data but increases the dynamic range of the chosen features so that they can be distinguished more easily from the background. Since the enhancement techniques are problem-specific, a technique that is suitable for one application may not be suitable for other applications. Two image enhancement techniques are useful in improving the quality of the images for boundary extraction stage: smoothing and sharpening.

#### **3.6.1 Background**

A digital image contains many spatial frequency components. Different frequency components characterize different features on the image. For example, the high frequency components correspond to the edges in an image. Many image enhancement techniques are based on a spatial operation performed on neighbouring pixel values rather than on an individual pixel value. Filters are algorithms that transform the original image into an enhanced image by attenuating certain frequencies and passing other frequencies. This filtering operation is done by convolution which applies a  $N \times N$  mask to every pixel in an image. That is, the pixel value depends on the pixel being processed and on the

surrounding pixels[20]. For example, in a two-dimensional image, the gray level of the centre pixel of a 3x3 mask will be replaced by reference to the gray levels of its eight neighbours as well as to its own original value. Using a larger mask provides more spatial information but increases the number of calculations per pixel which are proportional to the square of the size of the mask.

Special treatment is required when the convolution reaches the border of an image because the mask will extend beyond the boundary. For example, without special treatment a 3x3 mask will result in an enhanced image that is two lines and two columns smaller than the original image. Three techniques[15] can be applied to the image for processing data in the border region. The first one is to artificially extend the original image beyond its border by repeating the border pixel values. Then after the filtering process is completed, the extended portion of the image is discarded. The second technique is to replicate the average pixel values near the borders based on the image behaviour within a few pixels of the border. The third technique is to ignore the effect of the border pixel value in the overall image by retaining the original border pixel value in the output image. This is applicable only when the region of interest does not include a border region. In magnetic resonance images, the object of interest is generally positioned in the middle of the image; therefore, the third technique is adopted because little information is lost and the operation is simplified.

### 3.6.2 Smoothing by neighbourhood averaging

Neighbourhood averaging is a spatial-domain smoothing technique that replaces each point by the average value of its neighbours. It is a low-pass filtering operation which attenuates the high-frequency components while preserving the low-frequency components. Since noise contains high-frequency components, smoothing reduces the noise amplitude. However, edges are also composed of high-frequency components;

therefore, smoothing blurs or smears edges of the object and should be applied with care in order to preserve edges.

Neighbourhood averaging outputs the value of a particular pixel position by evaluating the input pixel value and the surrounding pixel values. That is, each pixel is replaced by a weighted average of its neighbourhood pixels [14]:

$$V(m,n) = \sum_{(k,l) \in w} a(k,l) y(m-k, n-l) \quad (3.7)$$

where  $y(m,n)$  and  $V(m,n)$  are the input and output images, respectively;  
 $w$  is a suitably chosen filter convolution mask;  
 $a(k,l)$  are the filter weights;

In order not to affect the overall brightness of an image, the convolution mask is divided by a scalar ( $s$ ) so that the sum of the filter weights is equal to 1:

$$V(m,n) = 1/s \sum_{(k,l) \in w} a(k,l) y(m-k, n-l) \quad (3.7.1)$$

This operation is repeated for every pixel in the input image and produces a smoothing effect.

Fourier analysis is used to determine the cut-off frequency of an ideal low-pass filter that will retain sufficient information on the bone structure. Figure 3.11(a) is the image of the skull and Figure 3.11(b) shows the Fourier spectrum of the image in Figure 3.11(a). The Fourier spectrum is log-scaled with super-imposed circles having radii of 99.4, 198.8, 298.5, and 398.4 cycle/meter, respectively. The selection of these radii is

based on equal distance between circles on the Fourier spectrum. The results of applying ideal low-pass filters with cut-off frequencies at the above radii are shown in Figure 3.12. Figure 3.12(a) does not provide enough information in identifying different structures of the skull because of the blurring effect of the low-pass filtering. The severe blurring indicates that most of the edge information is contained in the high frequency contents outside the cut-off frequency. As the radius of the filter increases, the degree of blurring decreases. Figure 3.12(c) and (d) provide sufficient information on the bone structure while the background low gray-level variation is smoothed out; therefore, any low-pass filters having a cut-off frequency greater than 298.5 cycle/meter will retain sufficient information on the bone structure.

After determining that cut-off frequency greater than 298.5 cycle/meter will provide enough information on the bone structure, an  $3 \times 3$  neighbourhood averaging mask of one's is tested for its usefulness in retaining bone information. Figure 3.13 Part (a) is the original image consists of a synthetic black circle and part (b) is the smoothed image. Part (b) and (d) are the corresponding Fourier spectrum. The Fourier spectrum is scaled to the range of 1 to 254 gray levels. Part (c) and (f) are the corresponding gray-level profile plots of the Fourier spectrum along the horizontal line passing through the centre. The cut-off frequency of the filter is determined by finding the half power point of the filtered image. By comparing the scaled power level in part (c) and (d), the half power point occurs at 335.6 cycle/meter. That is, the cut-off frequency of the neighbourhood averaging mask is 335.6 cycle/meter which is greater than 298.5 cycle/meter and retains sufficient information on the bone structure.

### 3.6.3 Sharpening by unsharp masking

The unsharp masking technique is used commonly in the printing industry for sharpening edges. The sharpening effect is produced by subtracting the blurred version

of the original image from the original image. In general, the unsharp masking operation can be represented by [14]:

$$\begin{aligned} g(x,y) &= f(x,y) - f'(x,y) \\ &= (f(x,y) \ominus h') - (f(x,y) \ominus b') \\ &= f(x,y) \ominus (h' - b') \end{aligned} \quad (3.8)$$

where  $g(x,y)$  and  $f(x,y)$  are the output and input image, respectively;  
 $f'(x,y)$  is the blurred version of  $f(x,y)$ .

$h'$  is a mask of :

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$b'$  is a local weighted averaging mask:

$$\frac{1}{8} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$\ominus$  = convolution.

That is, the image is convoluted by the mask in Table 3.3.

Table 3.3: Unsharp mask

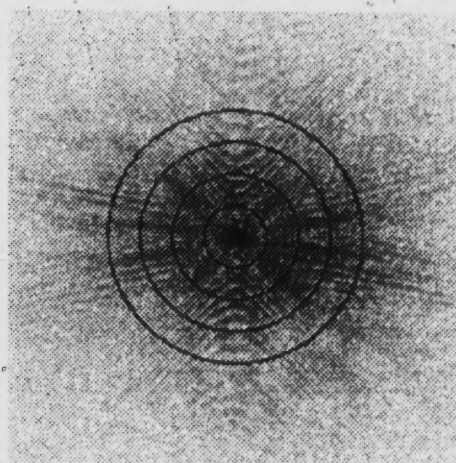
---


$$\frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$


---



(a)



(b)

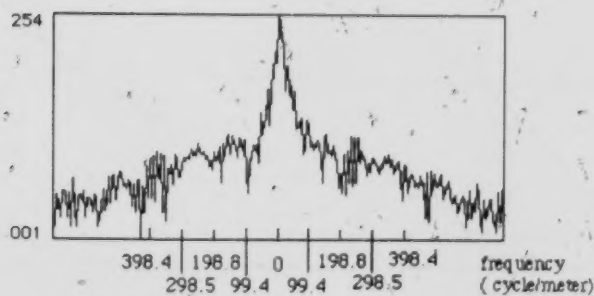


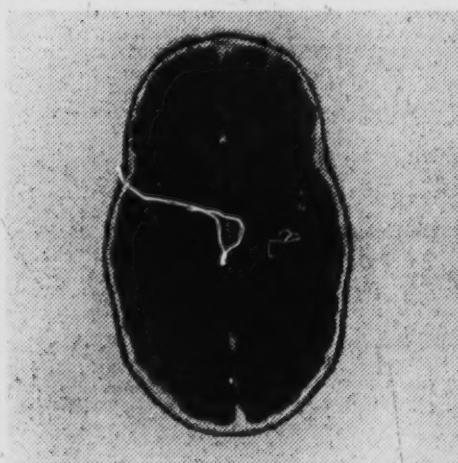
**Figure 3.11:** (a) magnetic resonance image.  
 (b) Fourier spectrum of (a). The super-imposed circles have radii equal to 99.4, 198.8, 298.5, and 398.4 cycle/meter, respectively.  
 (c) The gray-level profile plot along the horizontal line passing through the centre of (b).



(a)  
Cut-off frequency = 98.4 cycle/meter



(b)  
Cut-off frequency = 198.8 cycle/meter



(c)  
Cut-off frequency = 298.5 cycle/meter



(d)  
Cut-off frequency = 398.4 cycle/meter

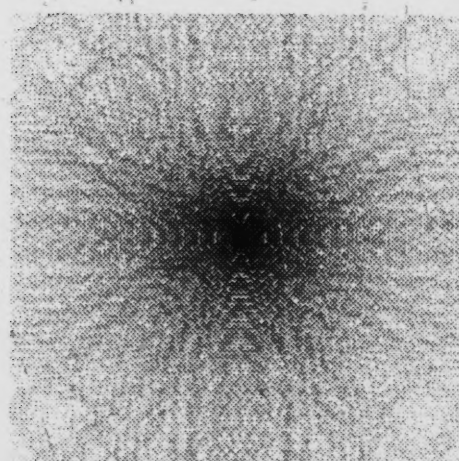
**Figure 3.12:** Results of applying ideal low-pass filtering to Figure 3.12(a). The radii shown in Figure 3.12(b) is used.



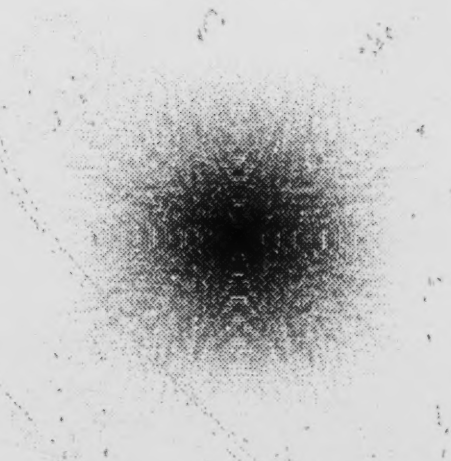
(a) Original image.



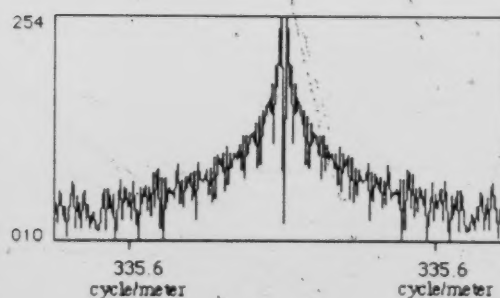
(d) Smoothed image.



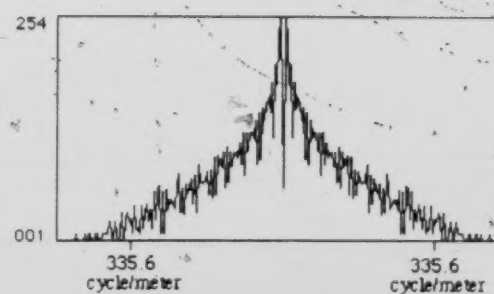
(b) Fourier spectrum of (a).



(e) Fourier spectrum of (d).



(c) Gray-level profile plot along a horizontal line passing through the center.



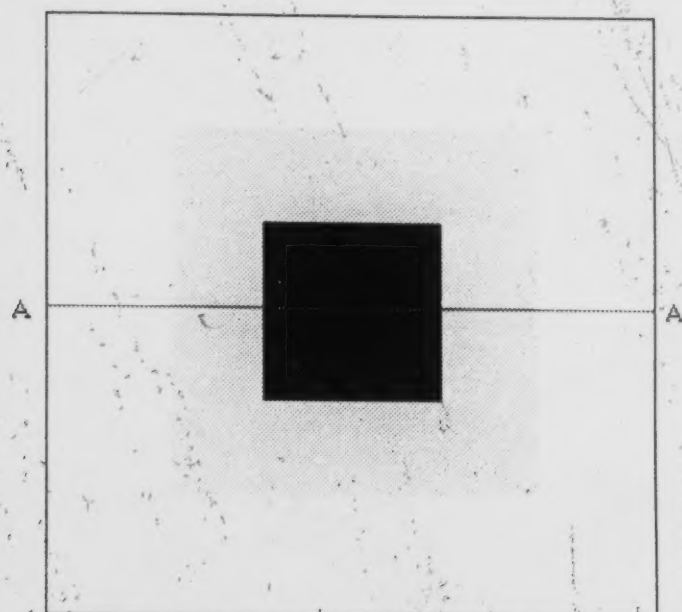
(f) Gray-level profile plot along a horizontal line passing through the center.

Figure 3.13: Example of image smoothing by neighbourhood averaging.

The effect of sharpening by the unsharp masking technique is demonstrated by convolving the mask ( Table 3.3 ) to the image in Figure 3.14(a). The test object consists of a black square surrounded by two square rings having pixel values of 30 and 20 respectively on a white background. Figure 3.14(b) is the gray-level profile plot along AA'. Figure 3.15 shows the result of applying the sharpening technique to Figure 3.14(a). Figures 3.15(a) and (d) are the original image and sharpened images, respectively. Figures 3.15(c) and (f) are the corresponding gray-level profile plots along the horizontal line passing through the centre of Fourier spectrum in part (b) and (e). Figure 3.15(d) shows that the edges are amplified by the sharpening operation. Comparing the scaled power level in Figures 3.15(c) and (f), the sharpening operation increases the magnitude of the high frequency components which provide the edge information. For example, the scaled power level at 360.6 cycle/meter for the sharpened image is 141 while for the same frequency in the original image is 115.

### 3.7 Boundary Data File

Although a number of three-dimensional reconstruction programs are commercially available, for example, Scan Protocol by Cemax and the image program by 3D Biomedical Imaging Inc., there is no standard file format for three-dimensional models. Also, a review of the literature is not helpful in providing information regarding any particular three-dimensional data file format. Therefore, a data format was designed to meet our requirements with minimum overhead. The boundary data file format consists of a 7-byte header followed by the direct coding of boundary coordinates



(a)

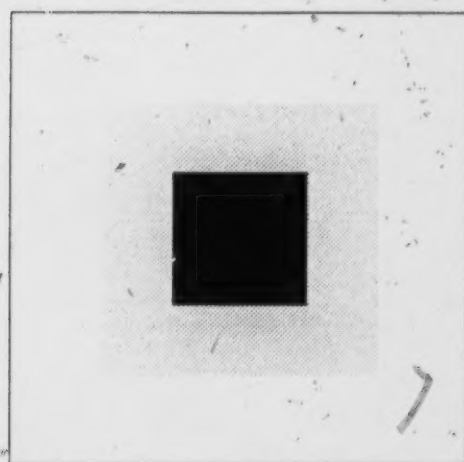


(b)

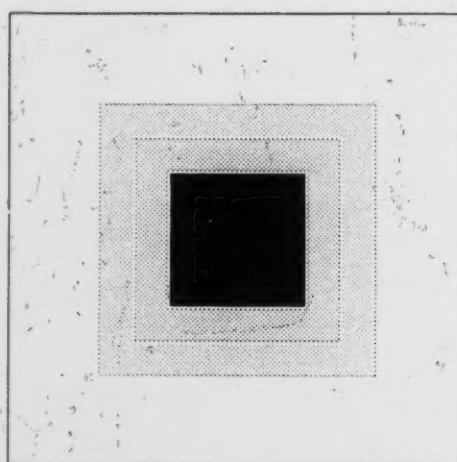
**Figure 3.14** Test image for sharpening by unsharp masking technique.

(a) Test object consists of a black square surrounded by two square rings.

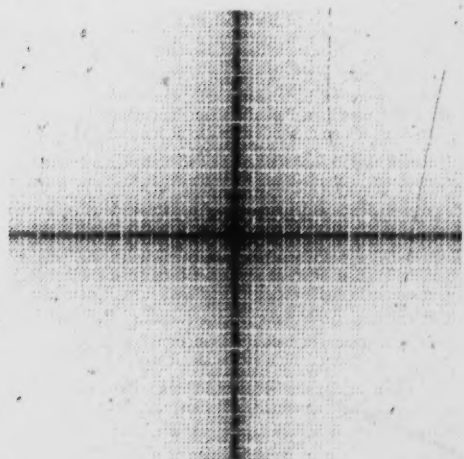
(b) Gray-level profile plot of the image along AA'.



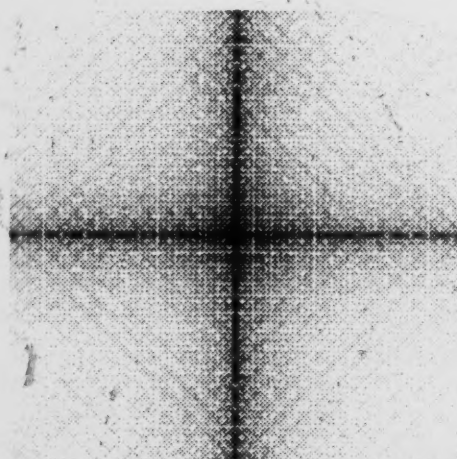
(a) Original image.



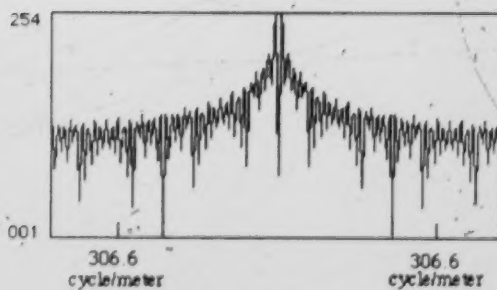
(d) Sharpened image.



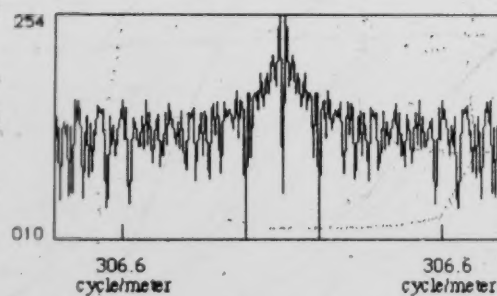
(b) Fourier spectrum of (a).



(e) Fourier spectrum of (d).



(c) Gray-level profile plot of the spectrum along the horizontal line passing through center.



(f) Gray-level profile plot of the spectrum along a horizontal line passing through the center.

Figure 3.15: Example of image sharpening by unsharp masking.

mentioned before. The header has the following format:

|           |                                                             |
|-----------|-------------------------------------------------------------|
| Byte 1,2  | the total size of the data file. ( maximum $2^{16} - 1$ )   |
| Byte 3    | the number of slices in the data file. ( maximum 20 )       |
| Byte 4    | the thickness of the slices in millimetres. ( maximum 255 ) |
| Byte 5, 6 | the size (in millimetres) of the number of pixels.          |
| Byte 7    | the inter-slice factor. ( maximum 255 )                     |

Bytes 3 to 7 are parameters of the magnetic resonance images which are used in calculating the physical dimension of the three dimensional model. All of the three dimensional data for a set of magnetic resonance imaging images is stored in the same file, separated by end-of-line markers and end-of-slice markers to distinguish between sections in an image and contours between slices respectively.

### 3.8 Three-Dimensional Reconstruction Process

'3-D' reads the three dimensional boundary data created by 'HEAD' and stores the data in a two dimensional array as shown in Figure 3.16. The two-dimensional array consists of a static header column with 60 rows for the boundary data. Each row contains the boundary points of a contiguous curve. That is, if the contour of a slice is broken up into three portions, each portion will occupy one row in this two-dimensional array. The static header column consists of a pointer to a data cell and the slice number that the contour belongs to. A data cell (shown in Figure 3.17) is a record of the x, y, and z coordinates of the boundary point, a visibility flag and an intersection flag, used in the hidden surface removal process, and a pointer to the next data cell. These data cells are dynamically allocated to reduce memory requirements during execution.

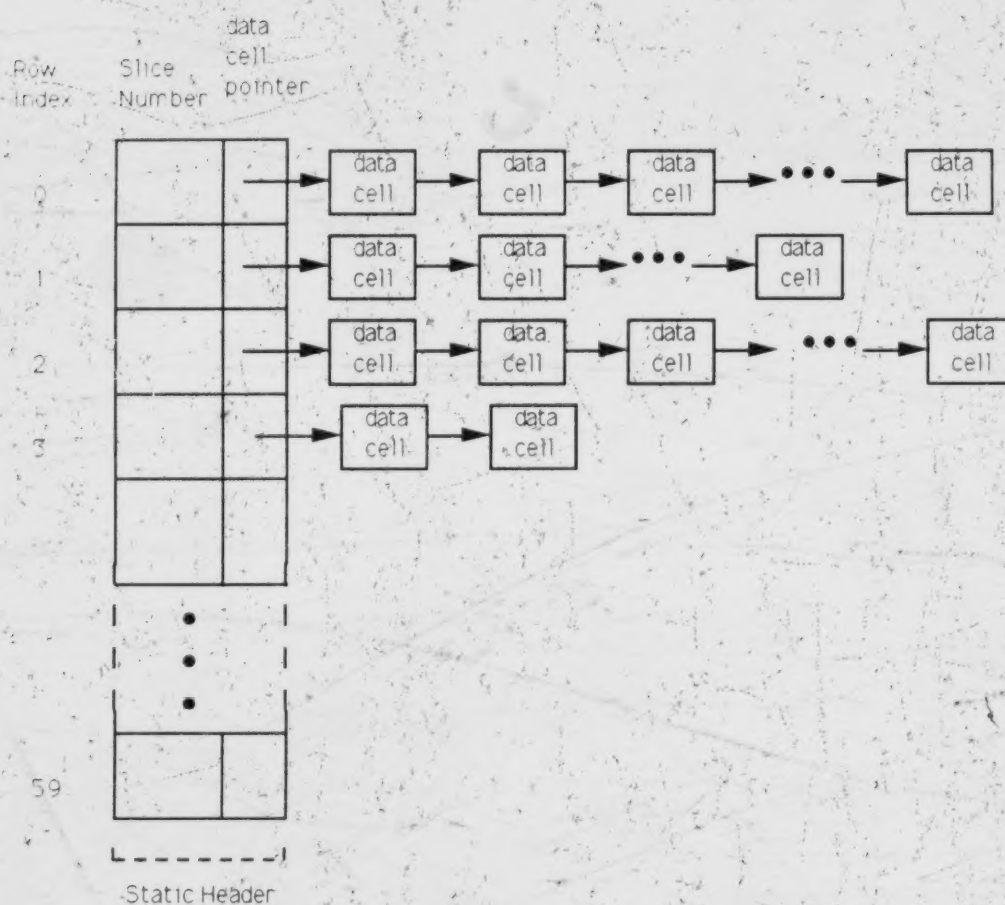


Figure 3.16: Three-dimensional model array.

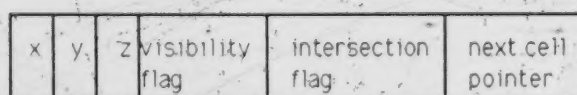


Figure 3.17: A data cell.

### 3.9 Hidden surface Removal

A wire frame display is produced by first transforming the end points of a line into the new coordinates. Then these end points are connected by a two dimensional line. This method is easy to produce and does not require a large amount of computation time. One of the main problems with wire frame displays is that the user can see through objects that should be solid. Therefore, a wire frame display with hidden surface removal gives a better impression of a three dimensional object. A Z-buffer hidden surface removal algorithm is based on the Z value of the end points in determining the visibility of lines. This method is simple and relatively efficient in computation time and can be applied to objects of different shapes.

A Z-buffer is a two dimensional array that has the same size as the display window and contains the Z-coordinate of the object. At the beginning of this algorithm, the Z-buffer is initialized to a large negative value (-9999). Then the 3-D data points are transformed into their new coordinates. As each data point is filled into the Z-buffer, its transformed Z value is compared to the Z value currently stored in the buffer. If the Z value of the data point is smaller than the Z value in the Z-buffer, the data point is hidden by a surface. In this case the visibility flag of the data cell is set to invisible and processing continues at the next data point in the 3D model. If the Z value of the data point is greater than the current value in the Z-buffer, the data point is visible and the visibility flag is set. If two data points are transformed to the same pixel location, the one which has a greater Z value is set visible and the other one is set invisible.

### 3.10 Model Display

After the hidden surface removal algorithm sets the visibility flag in each data cell, the visible portion of the contour is displayed in the 3D window by using draw line

routines in Graf3D library of the Macintosh MPW development tools. The display window has a size of 256x256 with six control scrollbars as shown in Figure 3.18. The six scrollbars are translation in upward/downward and left/right direction, rotation about x, y, and z axis with the angle shown under the scrollbars, and zooming in and out of the model displayed.

### 3.11 Conclusion

Image enhancement is useful in highlighting the regions of interest in an image. By using smoothing and sharpening techniques, an enhanced image is produced for feature identification or data extraction. Boundary extraction is useful in extracting a region of interest that has certain properties for identification. The extraction process consists of two steps: an edge detection step followed by a contour tracing step. Both the image enhancement process and the boundary extraction process are important in collecting data for the three dimensional reconstruction process. The quantitative measurements of edge detection operators show that Prewitt operator is the best; but the result is problem oriented. The edge detection operators may have a different performance when applied to objects with sharp corners like the rectangle. After the boundary data has been produced by 'HEAD', '3-D' performs a three dimensional reconstruction to create a three dimensional model. Six controls in manipulating the orientation of the 3-D model are provided by control scrollbars.

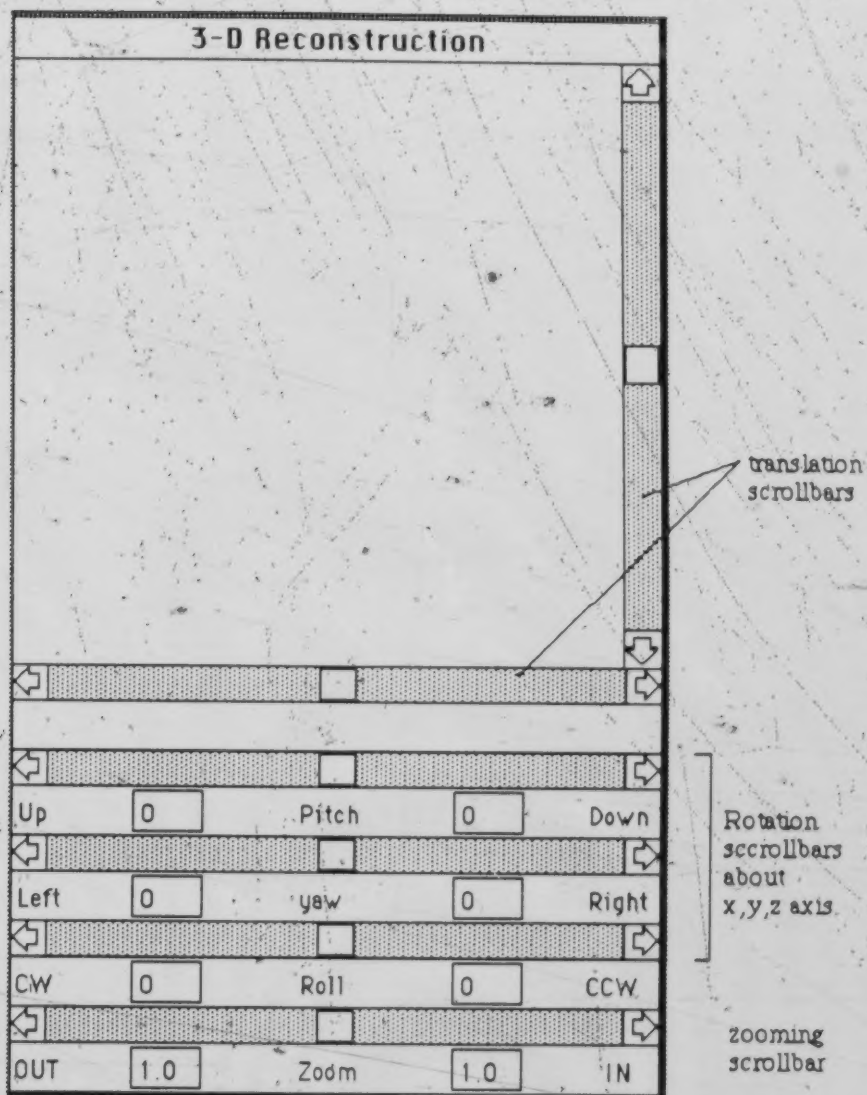


Figure 3.18: '3D' display window.

## References

- [1] Aldus Corp., TIFF Developer's  $\pi$ , 1988, Aldus Corporation, Seattle, WA., Nov. 1988.
- [2] Atkinson K.E., An Introduction to Numerical Analysis, 1978, Wiley, New York.
- [3] Ballard D.H., Brown C.M., Computer Vision, 1982, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.
- [4] Batnitzky S., Price H.I., Lee K.R., Cook, P.N., Cook, L.T., Fritz, S.L., Dwyer S.J., & Watts C., "Three-dimensional Computer Reconstructions of Brain Lesions from Surface Contours Provided by Computed Tomography: A Prospectus", Neurosurgery, Vol. 11, No. 1, 1982, pp. 73-84.
- [5] Batnitzky S., Price H.I., Lee K.R., Cook, P.N., Cook, L.T., Fritz, S.L., & Dwyer S.J., "Three-dimensional Computer Reconstructions from Surface Contours for Head CT Examinations", Journal of Computer Assisted Tomography, Vol. 5, No. 1, 1981, pp. 60-67.
- [6] Canny F.J., "Finding Edges and Lines in Images", M.I.T.A.I. Lab., Tech. Rep. AI-TR 720, June 1983.
- [7] Chakravapty I., "A single-Pass, Chain Generating Algorithm for Region Boundaries", Computer Graphics and Image Processing, Vol. 15, 1981, pp.182-193.
- [8] David L.S., "A Survey of Edge Detection Techniques", Computer Graphics and Image Processing, Vol. 4, 1975, pp. 248-270.
- [9] Freeman H., "Computer Processing of Line Drawing Images", Computer Surveys, Vol. 6, 1974, pp.57-97.
- [10] Fu K.S., Mui J.K., "A Survey on Image Segmentation", Pattern Recognition, Vol. 13, 1981, pp. 3-16.
- [11] Gonzalez R.C., Wintz P., Digital Image Processing, 1977, Addison-Wesley Publishing Company.
- [12] Haralick R.M., "Digital Step Edges from Zero Crossing of Second Directional Derivatives", IEEE Trans. Pattern Anal. Machine Intell., Vol. PAMI-6, Jan. 1984, pp. 58-68.

- [13] Henrich G., Mai N., and Baekmund H. "Preprocessing in Computed Tomography Picture Analysis: A Bone-Deleting Algorithm," *Journal of Computer Assisted Tomography* 3(3): 379-384, June, 1979.
- [14] Jain/A.K., *Fundamentals of Digital Image Processing*, 1989, Prentice-Hall, New Jersey.
- [15] Jensen J.R., *Introductory Digital Image Processing*, 1986, Prentice-Hall, New Jersey.
- [16] Keller J.M., Edwards F.M., and Rundle R. "Automatic Outlining of Regions on CT scans", *Journal of Computer Assisted Tomography*, 5(2):240-245, April 1981.
- [17] Kulkarni A.K., "Sequential Shape Feature Extraction from Line-Drawings", *Proc. IEEE Computer Society Conference, PRIP, Chicago, Illinois*, May 1978, pp.230-237.
- [18] Liu H.K., "Two- and Three-dimensional Boundary Detection", *Comput. Graphics and Image Processing*, Vol. 6, pp.125-134, 1977.
- [19] Levialdi S., "Finding the edge", *Digital Image Processing*, Simon J.C. and Haralick R.M. ed., pp.105-148, D. Reidel Publishing Company, Dordrecht, Holland, 1981.
- [20] Loebl J., *Image Analysis: Principles and Practice*, 1985, A Vickers Company.
- [21] Peli T., Malah D. "A Study of Edge Detection Algorithms", *Comp. Graphics and Image Processing*, Vol. 20, 1982, pp. 1-21.
- [22] Pratt W.K., *Digital Image Processing*, 1977, Wiley, New York.
- [23] Prewitt, J.M.S. "Object Enhancement and Extraction," *Picture Processing and Psychopictorics*, New York: Academic Press, Lipkin B.S. and Rosenfeld A. (eds), 1970
- [24] Shaw G.B., "Local and Regional Edge Detectors: Some Comparisons", *Computer Graphics and Image Processing*, Vol. 9, 1979, pp. 135-149.
- [25] Udupa J.K., "Display and Analysis of 3D Medical Images Using Directed Contours", *Proc. NCGA 6th Ann. Conf.*, Dallas.

## CHAPTER 4: SOFTWARE APPLICATION

### 4.1 Introduction

The Macintosh is an event-driven computer. A great deal of its time is spent in an endless loop, waiting for the user to trigger it into action. Viewed from the uppermost level, the main event loop in the Macintosh programs is a layered set of routines, as shown in Figure 4.1[2]:

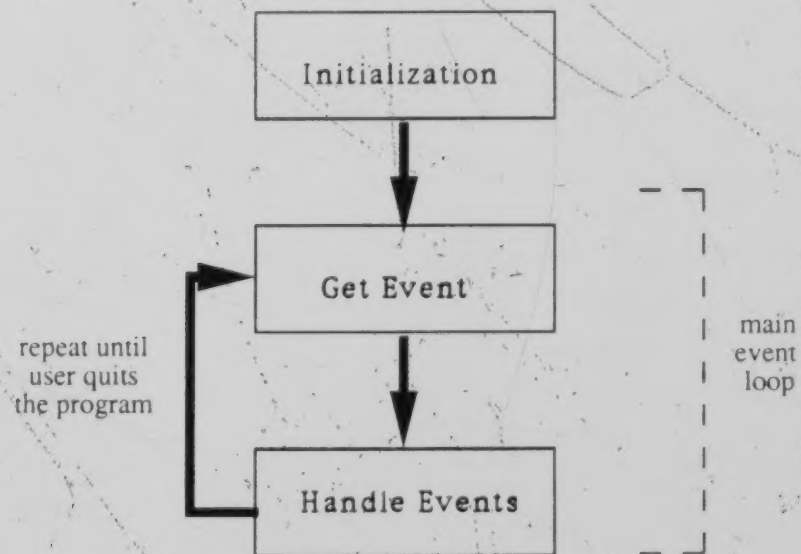


**Figure 4.1:** The main event loop of Macintosh programs.

When an application is invoked, it executes a series of initialization procedures to set up the environment in which the application runs. Then the application transfers control to the main event loop to wait and handle user initiated events. The main event loop consists of a *GetNextEvent* routine call followed by a series of conditional clauses that identify the type of event involved and processes it accordingly. If an event is found, *GetNextEvent* routine completes an **EventRecord** variable with the type of event and other information that depends on the event type, such as the location of a mouse click or a pointer to a window to be updated. When the event involves the user quitting the application, the loop ends, the application terminates, and the user is returned to MultiFinder.

A structure chart shows the relationship between program units without including any information about the order of activation of these units[3]. It is drawn using the following symbols:

1. a plain rectangle annotated with the name of a function,
2. a bold rectangle annotated with the name of a Toolbox routine,
3. an arrow connecting rectangles indicates the direction of function call,
4. a dashed arrow annotated with the data passed to and from elements in the structure chart,
5. a comma ',' between passing data indicates that the data are passed together in the function call; a backslash '/' between passing data indicates that only one of the data items is passed in the function call,
6. a bracket with figure number '(Figure x)', indicates the structure chart continues in Figure x.

Figure 4.2 shows the general form of the structure chart used in this chapter to describe the application software.

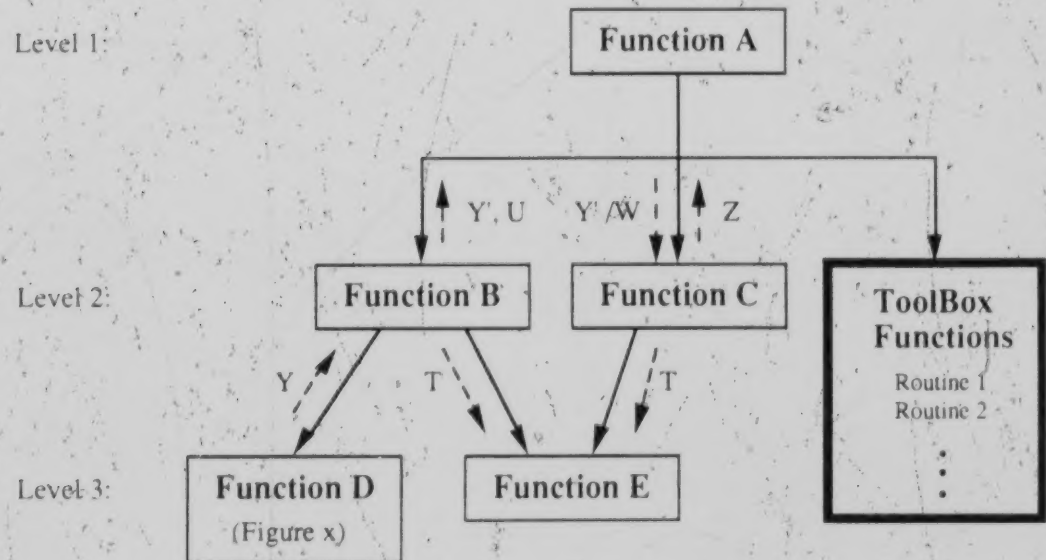


Figure 4.2: Structure chart format.

Function calls are from higher to lower levels in the structure chart. In Figure 4.2, Function A calls on Functions B, C and ToolBox Functions. Function B calls on Functions D and E. Nodes at level  $n$  in the chart can be shared by two or more nodes at level  $n-1$ . For example, Function E in level 3 is called by both Functions B and C in level 2. The left-to-right ordering of Functions and Routines does not imply that they are called in that sequence. Functions can be called more than once; each call may pass

different data of the same type. For example, Function A calls Function C twice, once with data Y' and once with data W. Parameters passed to Toolbox Routines are not shown in the structure chart; specific information regarding parameters and functions of these Toolbox Routines are defined in Inside Macintosh[1].

#### 4.2 'HEAD' - Boundary Extraction Software

The application 'HEAD' is the software program that performs boundary extraction of magnetic resonance images. It requires two megabytes of memory to execute its operations. When the application is invoked, 'HEAD' requires the user to enter the magnetic resonance imaging parameters for each set of magnetic resonance images (image data set). It reads in the image file in TIFF file format and creates two image windows to display the image. It produces a RESULT window on the lower left hand corner of the screen to display the x-y coordinates and gray-level of the mouse position in the image window. It provides operations to enhance and edit images so that the result is more suitable for boundary extraction. If the user selects auto-process, 'HEAD' will record the sequence of filtering operations applied to the first image file and automatically open the next image file and apply the same sequence to the image file until the end of the image data set. During auto-process, the user can stop the application of the stored sequence of filtering operations at any image files and perform manual intervention. If auto-process is not selected, the user needs to open the image files one at a time and manually select the filtering operations for each image file.

'HEAD' has the same event loop structure as described earlier in the introduction and the structure chart is shown in Figure 4.3. When the application is invoked, it first executes routines to initialize various system managers of the Macintosh operating system, to set up menus, to allocate memory, to process MultiFinder startup information

and to load in the required resources. Then it enters the main event loop waiting for the user to initiate events by either clicking the mouse or typing on the keyboard. After identifying the type of event, it handles the event according to the instructions associated with the event in the program. It executes the main event loop and responds to the user's input until the user quits the application.

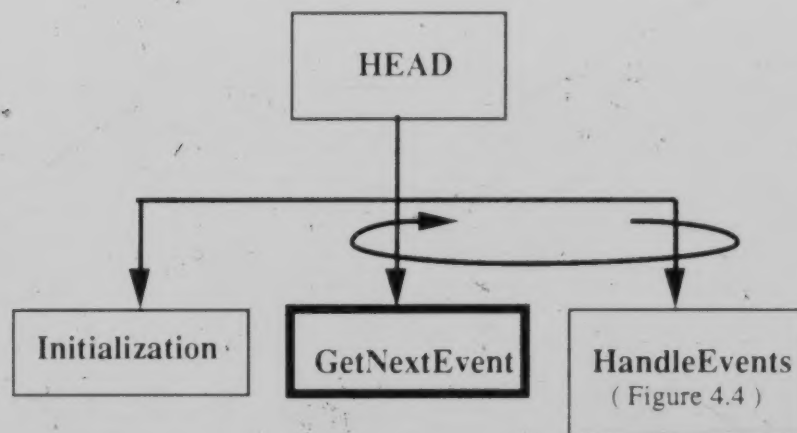


Figure 4.3: Structure chart of HEAD.

#### 4.2.1 Description of Software Routines

The following is a description of the subroutines implemented in 'HEAD'.

##### 1. Initialization (Figure 4.3)

This routine sets up the global environment for running the application. It first checks whether there is enough memory available to run the program. Secondly, it

initializes various Toolbox managers before the Toolbox functions can be called. Thirdly, it loads in the resources from the resource file and sets up the menu bars. Lastly, it resets all global variables and creates an offscreen port for data manipulations during the operation process.

## 2. HandleEvents (Figure 4.4)

This function checks the occurrence of an event by a Toolbox routine called *GetNextEvent*. If an event occurs, it determines the kind of event and handles the event according to the associated instructions. If the user quits the application, the **finished** flag is set to signal the termination of the application.

## 3. DoKeyDown (Figure 4.4)

This function checks if the Command key is pressed. If yes, the user selects a command through the key board. After recognizing the Command key, this function calls *DoMenuEvent* to deal with the user request.

## 4. DoMouseDown (Figure 4.4)

This function responds to a mouse click in the window region. The mouse position may be in the menu bar, system window, contents of the window, or the go-away box of the window. After identifying the source of the mouse event, the associated subroutines are called to process the user request.

## 5. DoActivate (Figure 4.4)

This function is called when the user switches between different windows. It deactivates the previous active window and activates the selected window by drawing the contents of the window.

#### 6. DoUpdate (Figure 4.4)

This function updates the contents of a window. The update event originates from user activities, program displays, or system-generated overlays of dialog boxes. Windows are checked front to back so that the active window is checked first and updated if necessary. Then the window behind the active window is checked, and so forth.

#### 7. DoMenuEvent (Figure 4.4 and 4.5)

This function processes the selection of a menu item. There are three menu bars: File menu, Edit menu, and Functions menu. After determining the selected menu item, this function calls the corresponding menu handling routines.

#### 8. DoMouseDownInWindow (Figure 4.4)

This function checks whether the mouse click is in the window region. If it is, it will redraw the window. If 'Eraser' is selected, it will erase the image value under the cursor and set it to the background value.

#### 9. CloseAllWindows (Figure 4.4)

This function closes all of the windows opened by this application and disposes of the pointers that are used in creating the windows.

#### 10. UpdatePicWindow (Figure 4.4, 4.9, 4.11 and 4.14)

This function updates the active window by copying the offscreen port to the screen port. It uses the Toolbox routines *CopyBits* for copying data from source to destination memory location.

#### 11. DoObject (Figure 4.4)

This function selects a cursor symbol according to the tools selection.

#### 12. DoErase (Figure 4.4)

This function performs the application of an eraser to the image.

#### 13. DrawObject (Figure 4.4)

This function erases the portions of image under the eraser cursor to background color.

#### 14. DoDigitize (Figure 4.5 and 4.6)

This function corresponds to the NEW SET menu item. It asks the user to enter the magnetic resonance image information. This information includes the number of slices in a set of magnetic resonance imaging scans, the type of image data, and the parameters of magnetic resonance imaging scans. It also allows the user to select the auto-process feature.

#### 15. GetFile (Figure 4.5 and 4.7)

This function corresponds to the OPEN menu item. It first displays a standard file dialog box to let the user select an image file to open. If the file exists, the selected file is read in and two image windows are opened side by side. The left-hand window is the working window to which all the operations are applied. The right-hand window is the reference window of the original image. Then the function sets up the REVERTTOSAVED and UNDO functions by storing the image in two buffer areas: **OrgImage** and **LastImage**.

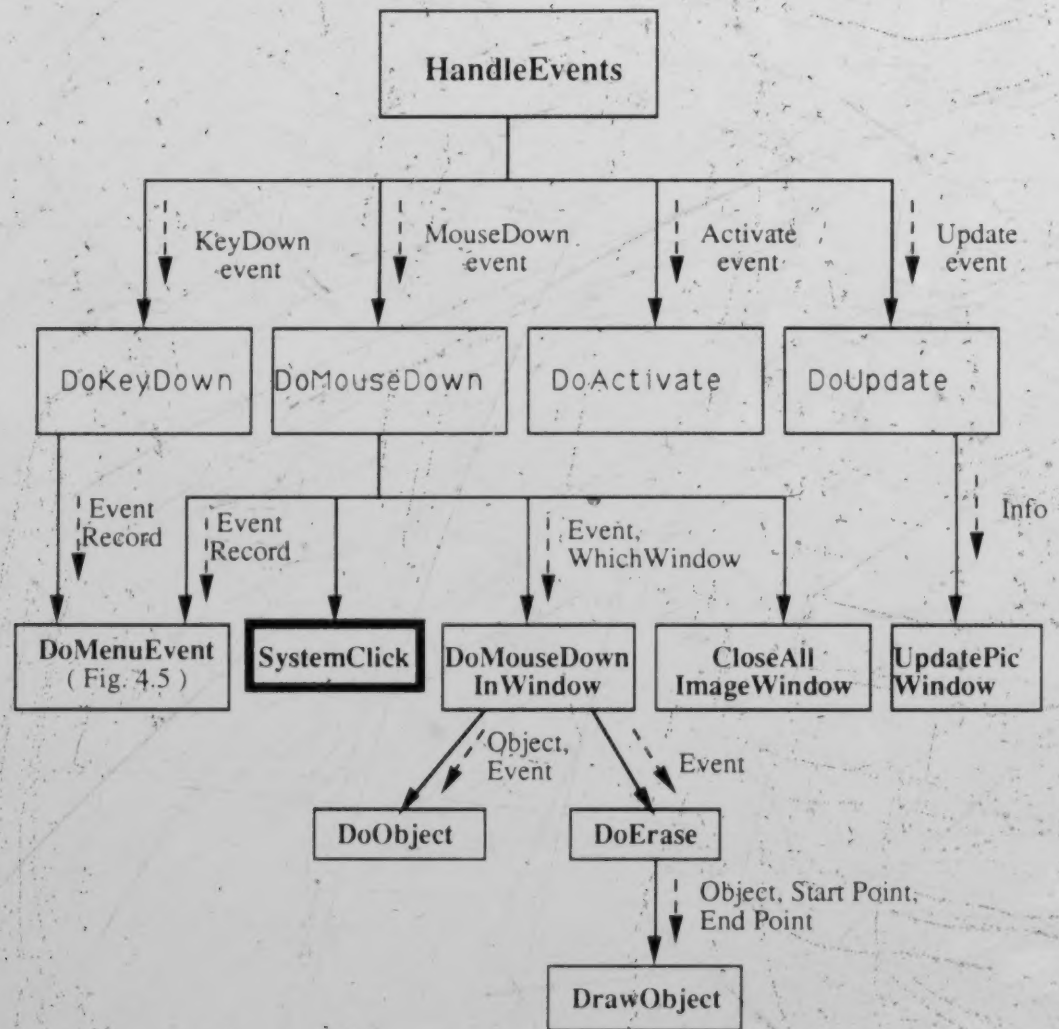


Figure 4.4: Structure chart of HandleEvents.

16. DoClose (Figure 4.5 and 4.8)

This function is called when CLOSE menu item is selected. It closes the currently active window, releases memory space associated with the window, and resets the menu items.

17. RevertToSaved (Figure 4.5 and 4.9)

This function erases all the filtering and editing operations performed on the image by restoring the original read-in image stored in **LastImage**.

18. DoSaveEdge (Figure 4.5 and 4.10)

This function first asks the user to enter a name for the new boundary data file. Then it writes the header information onto the new file before saving the boundary data. Before closing the data file, the size of the file contained in the header information is updated.

19. ProcessDig (Figure 4.5, 4.11 and 4.16)

This function applies the filtering operations to the image. The input parameter is the filtering type. After identifying the filtering operations, the corresponding code is executed and the result is stored in the offscreen port. Update of the new image is performed by *UpdatePicWindow*.

20. DoOutline (Figure 4.5 and 4.12)

This function implements the edge tracing algorithm described in Section 3.4. The boundary data is temporarily stored in a buffer area before saving to a file.

#### 21. DoEdge (Figure 4.5, 4.13 and 4.16)

This function implements the Prewitt edge detection algorithm described in Section 3.4. It uses a dialog box to ask for the threshold value. Then it invokes *ProcessDig* to perform the algorithm and produces a binary image.

#### 22. DoSetSeed (Figure 4.5)

This function allows the user to select the region of the boundary for contour tracing step. The seed point is selected by clicking the mouse at the region of interest. Multiple seed points are stored in an array for the boundary tracing process.

#### 23. DoAutoTrace (Figure 4.5)

This function controls the data extraction step. It first asks the user to enter the new file name. Then it initializes the header of the file before searching for the starting point. After the starting point is selected, the contour will be traced and the result is displayed on the screen. The user decides whether he/she is satisfied with the extracted contour data. If the user selects the **SAVE** item, the boundary data will be saved to a file using the boundary representation discussed in Section 3.5.

#### 24. AutoProcess (Figure 4.5 and 4.16)

This function applies the recorded filtering operations to all the images in an image data set. After finished working on one image file, it will automatically open the next image file until the end of the image data set.

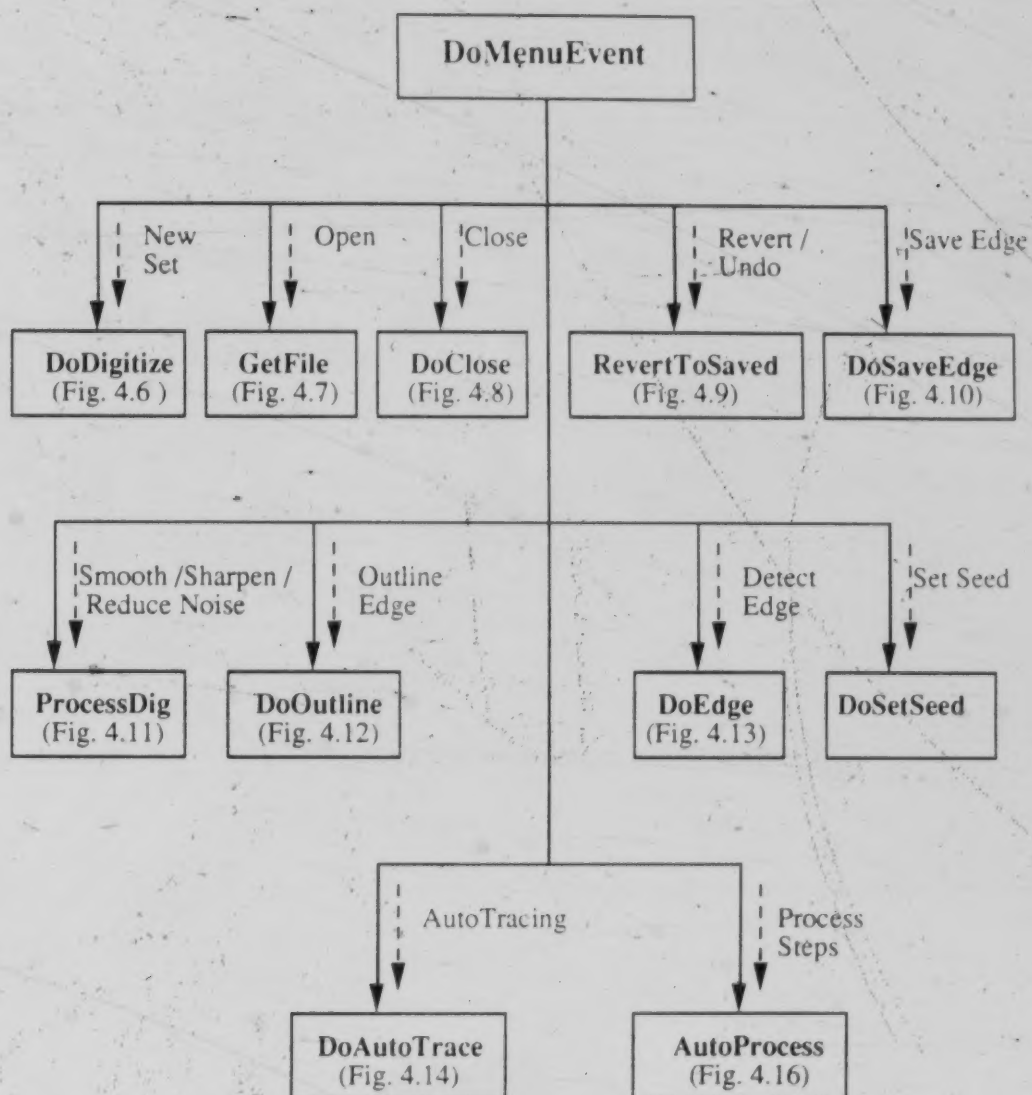


Figure 4.5: Structure chart of DoMenuEvent.

#### 25. EnterSlices (Figure 4.6)

This function displays a dialog box for the user to enter the number of slices in a set of magnetic resonance imaging scans. The input data is stored in the global variable **NumOfSlices**.

#### 26. SelectDataTypes (Figure 4.6)

This function displays a check box to allow the user to select the type of data source. There are two types of data source: MRI or CT. The input data is stored in a global variable **dType**. Only MRI data type is implemented and the selection of CT data type has the same effect as the selection of MRI data type.

#### 27. EnterPixelValues (Figure 4.6)

This function displays a dialog box to allow the user to enter the thickness of the slices in millimetres, the size of the pixel in millimetres over the number of pixels and the inter-slice factor. The input data is stored in global variables **sThick**, **pSize**, **pixel**, and **inFactor**.

#### 28. SelectAutoProcess (Figure 4.6)

This function displays a dialog box to allow the user to select auto-processing of the image data set. The selection is stored in variable **AutoStep**.

#### 29. OpenDigFile (Figure 4.7 and 4.16)

This function first checks whether there is enough memory available to read the image file by calling subroutine *GetMemory*. If sufficient memory is available, it creates two image windows and displays the image.

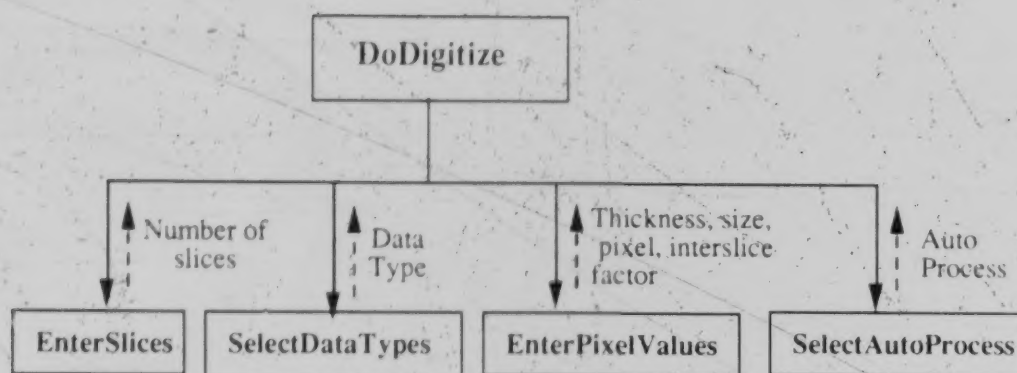


Figure 4.6: Structure chart of DoDigitize.

30. GetImage (Figure 4.7, 4.9, 4.12, 4.15 and 4.16)

This function reads in the image pointed to by **PicBaseAddr** of the window record into a 256x256 array.

31. GetMemory (Figure 4.7)

This function checks the size of free memory available to the application by calling the Toolbox routine *CompactMem*. If enough memory is available, the function returns the requested memory space; otherwise, it prints out an error message.

32. MakeNewWindow (Figure 4.7)

This function creates a window of the size of 256x256 pixels. It also sets up the window record of the new window.

33. CloseAwindow (Figure 4.8)

This function closes the active window and disposes all the pointers associated with the window.

34. GetFileName (Figure 4.10 and 4.15)

This function displays a standard file dialog box and waits for the user to enter the name of the boundary data file. If the user enters an existing file name, the new file name and the volume reference number to this file will be stored in global variable **gFileName** and **vRefNum**.

35. StoreData (Figure 4.10 and 4.15)

This function stores the boundary data in the boundary data file. After completing the writing to the file, the size of the data file is updated.

36. InitDataFile (Figure 4.10 and 4.15)

This function opens a boundary data file selected by the user and sets up the header of the file by storing the parameter information entered in NEW SET command.

37. GetRawLine (Figure 4.11)

This function copies a row from the *PicBaseAddr* to an array for filtering processing.

38. PutResultLine (Figure 4.11 and 4.14)

This function copies a row from an array to the *PicBaseAddr*.

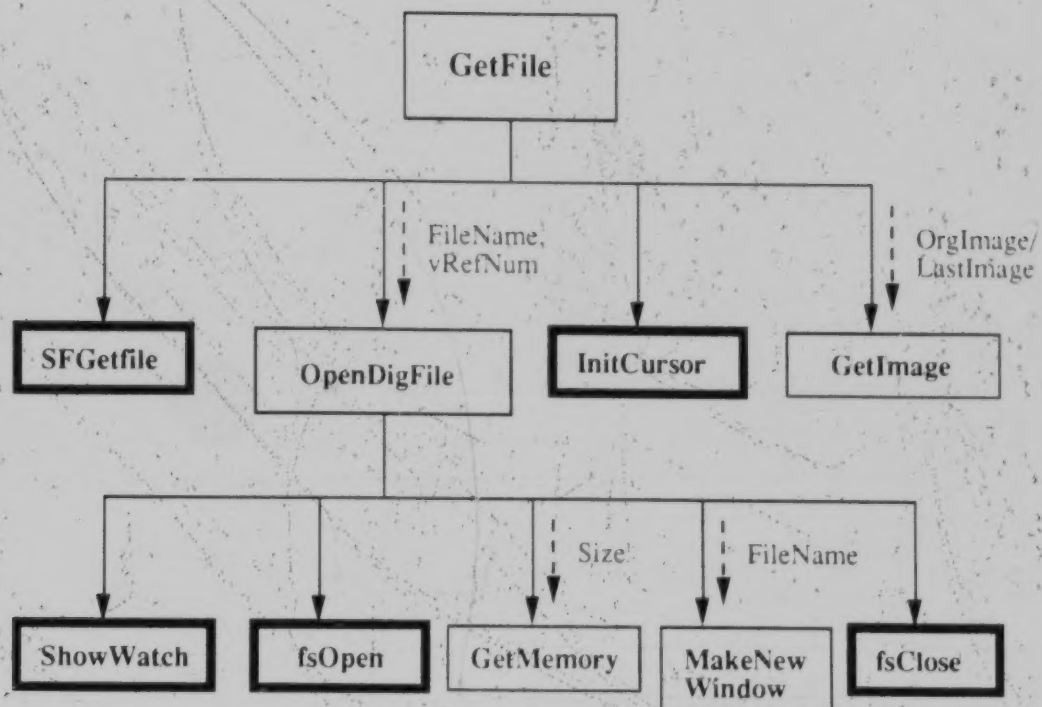


Figure 4.7: Structure chart of GetFile.

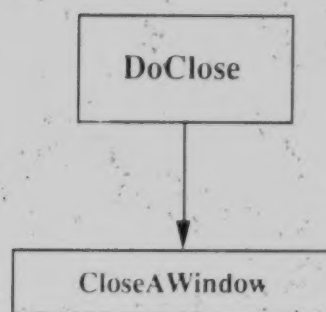


Figure 4.8: Structure chart of DoClose.

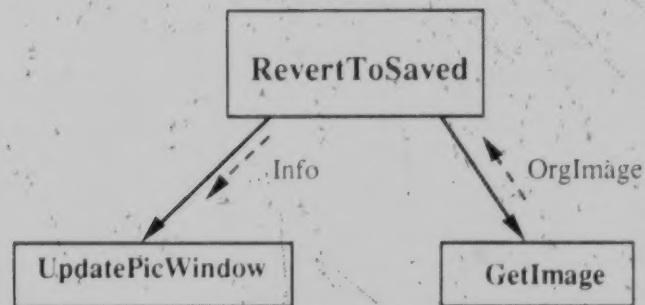


**Figure 4.9:** Structure chart of **RevertToSaved**.

#### 39. ContourTracing (Figure 4.12, 4.14, 4.15 and 4.16)

This function traces the boundary of an object and displays the extracted boundary data. It first searches for the starting point on the contour. While tracing the boundary of the object, it stores the boundary information in a buffer area which will be written to a file if the user is satisfied with the result of this function.

#### 40. EnterThresholdValue (Figure 4.13)

This function displays a dialog box and asks the user to enter the threshold value for filtering operations.

#### 41. ProcessDig (Figure 4.13)

This function applies the filtering operations to the image. The input parameter is the filtering type. After identifying the filtering operations, the corresponding code is executed and the result is stored in the offscreen port. Update of the new image is performed by *UpdatePicWindow*.

42. SetStartPoint (Figure 4.14)

This function records the seed points selected by the user through the mouse.

43. SearchStartPoint (Figure 4.15 and 4.16)

This function searches for an valid starting point on the contour as discussed in section 3.4(c).

44. GetConFile (Figure 4.16)

This function controls the sequence of auto-processing the images in a set of input data.

45. DoAutoOutline (Figure 4.16)

This function traces the contour of the object and displays the contour. A dialog box is displayed to allow the user to accept the extracted contour. If the user is not satisfied with the result of auto-process, the extracted data will be discarded and the original image is restored.

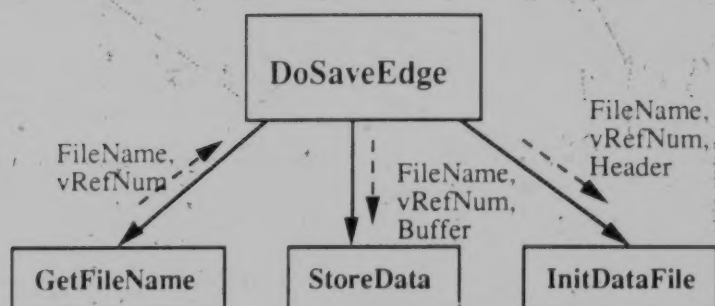


Figure 4.10: Structure chart for DoSaveEdge.

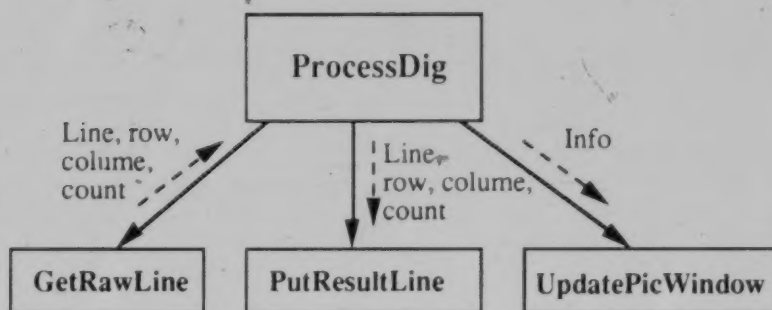


Figure 4.11: Structure chart of ProcessDig.

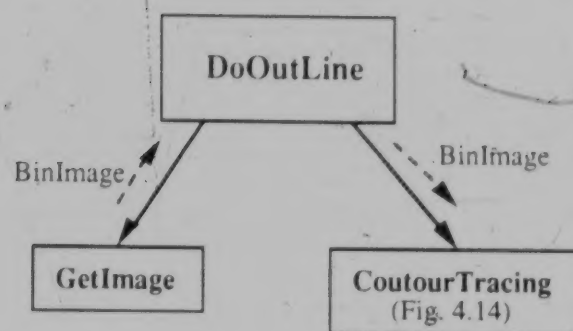


Figure 4.12: Structure chart of DoOutline.

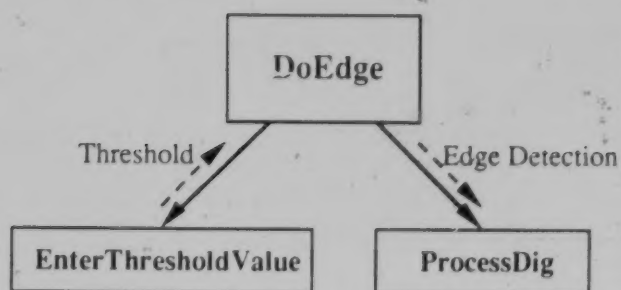


Figure 4.13: Structure chart of DoEdge.

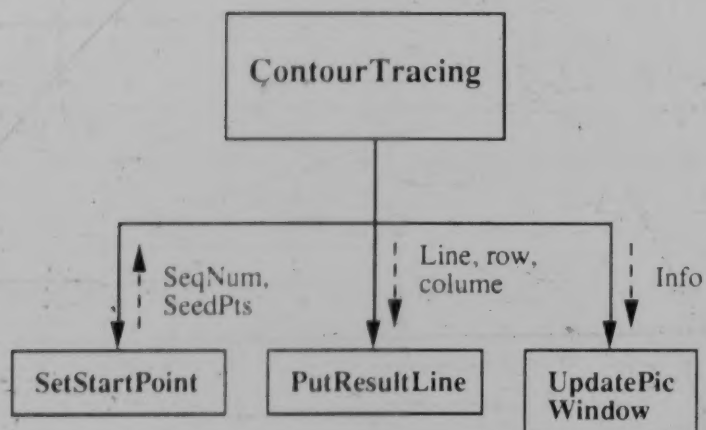


Figure 4.14: Structure chart of ContourTracing

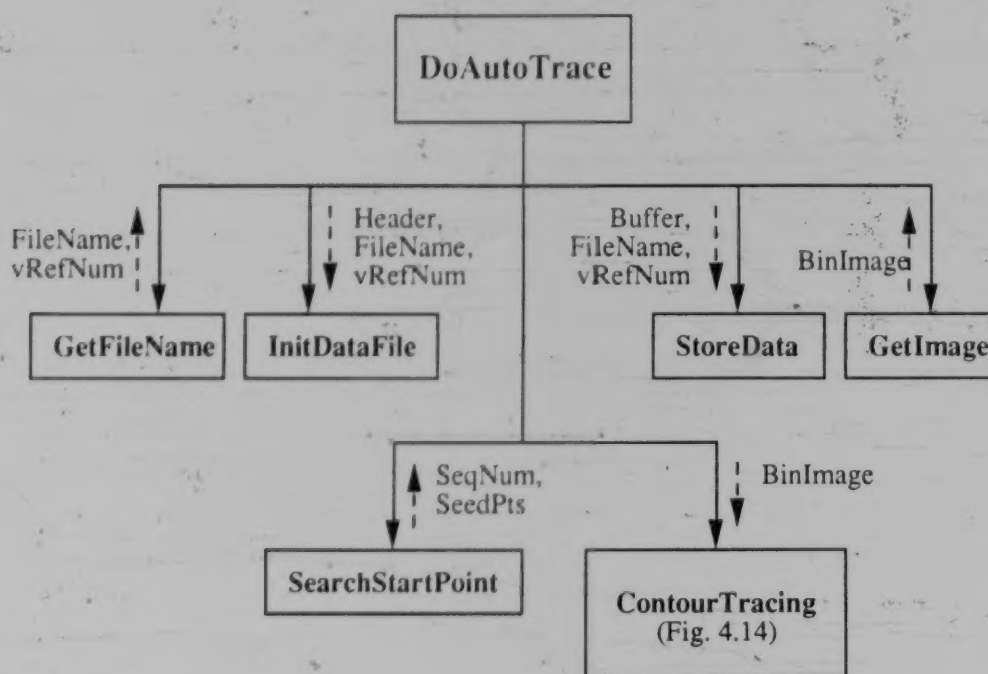


Figure 4.15: Structure chart of DoAutoTrace.

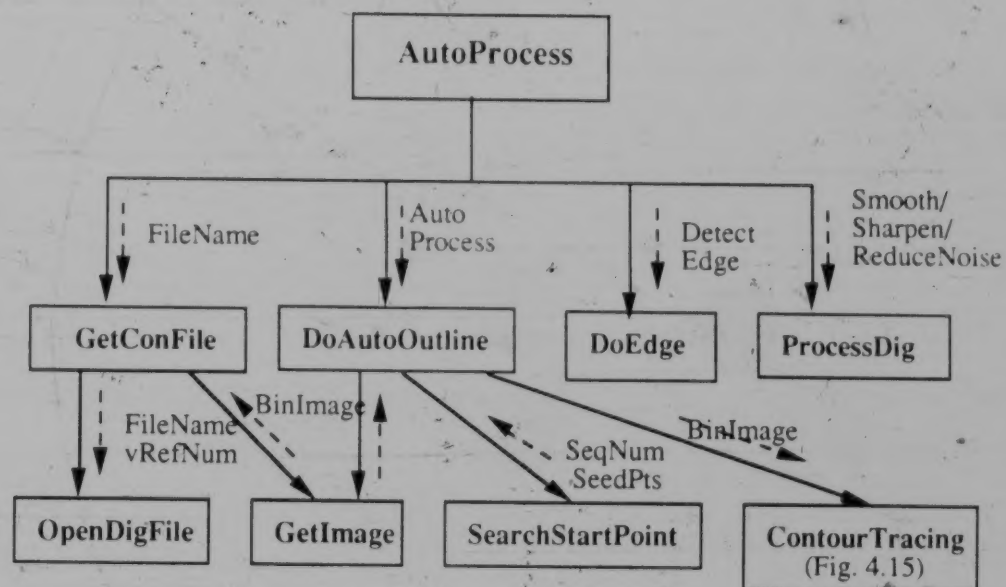


Figure 4.16: Structure chart of AutoProcess.

#### 4.2.2 Menu Command

'HEAD' has three menu bars: File Menu, Edit Menu, and Function Menu. Each menu item is assigned a command-key for quick selection in addition to the use of the mouse in the menu item selection. The characters assigned to the command-key may not correspond to the name of the menu item, but they are selected so that the command-keys are controlled by the left hand only which allows the user to free the right hand for using the mouse.

##### File Menu

The File menu includes operations for the current image file or the current image data set. It has the structure as shown in Figure 4.17.

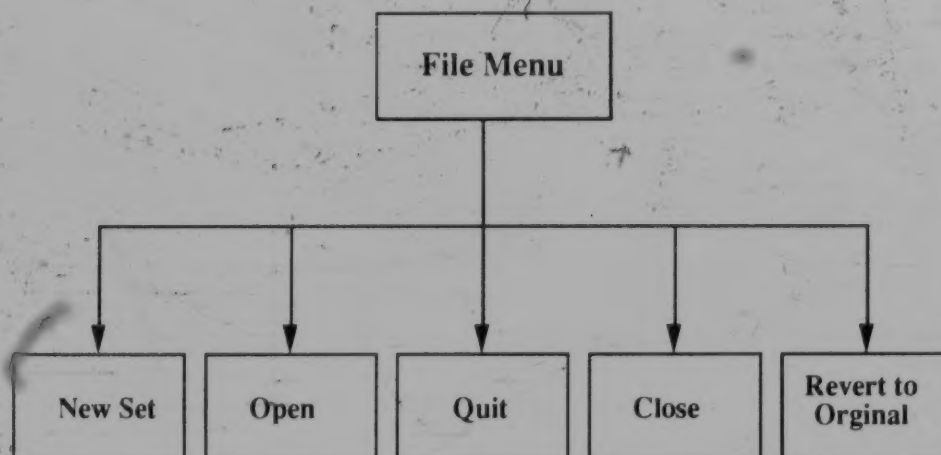


Figure 4.17: Structure of File Menu.

### 1) New Set ( command-N )

This function must be selected before any other functions can be invoked. It displays four dialog boxes consecutively and requires the user to enter or select the data for each set of magnetic resonance scans. These data are used in the three-dimensional reconstruction process. The first dialog box asks for the number of slices in a set of images. The second dialog box asks for the type of input images. The third dialog box asks for the slice thickness of magnetic resonance imaging slices, the size of a pixel over the number of pixels, and the inter-slice factor. The fourth dialog box queries the user regarding the automatic processing of the image data set.

### 2) Open ( command-O )

This function opens an image file. A standard file dialog box is displayed on the screen asking for the existing file to open. After the selection of the file name, the image file is read in and two image windows are created. The left window is a working window in which the manipulation of images is performed and the right window is a reference window which displays the original image. Only one image file can be opened at a time. After opening a file, this menu item will be disabled until the image windows are closed.

### 3) Close ( command-W )

This function closes the two active image windows on the screen. No changes on the working window will be saved into the original image file.

### 4) Revert to Original ( command-R )

This function undoes all the changes made since the file is opened by reverting the image to the original read-in version.

### 5) Quit ( command-Q )

This function closes all windows created by the application; stops the execution of the application, and returns the user to the MultiFinder.

### Edit Menu

The Edit menu provides operations on the current image in the working window. Its structure is shown in Figure 4.18.

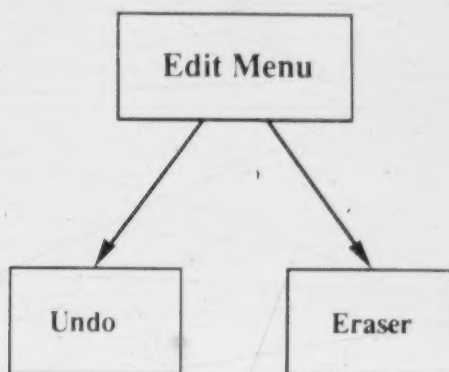


Figure 4.18: Structure of Edit Menu.

### 1) Undo ( command-Z )

This function undoes the last operation for the current window by restoring the version of the image before the last operation applied to the image.

## 2) Eraser ( command-E )

This function erases portions of the image under the eraser icon to the background color. After pointing the icon to the desired location, the portion of the image under the icon is erased by clicking the mouse. For erasing a continuous region, the mouse is held down while the icon is moved over the region.

## Functions Menu

The Functions menu provides enhancement and boundary extraction operations described in Sections 3.5 to 3.8. Its structure is shown in Figure 4.19.

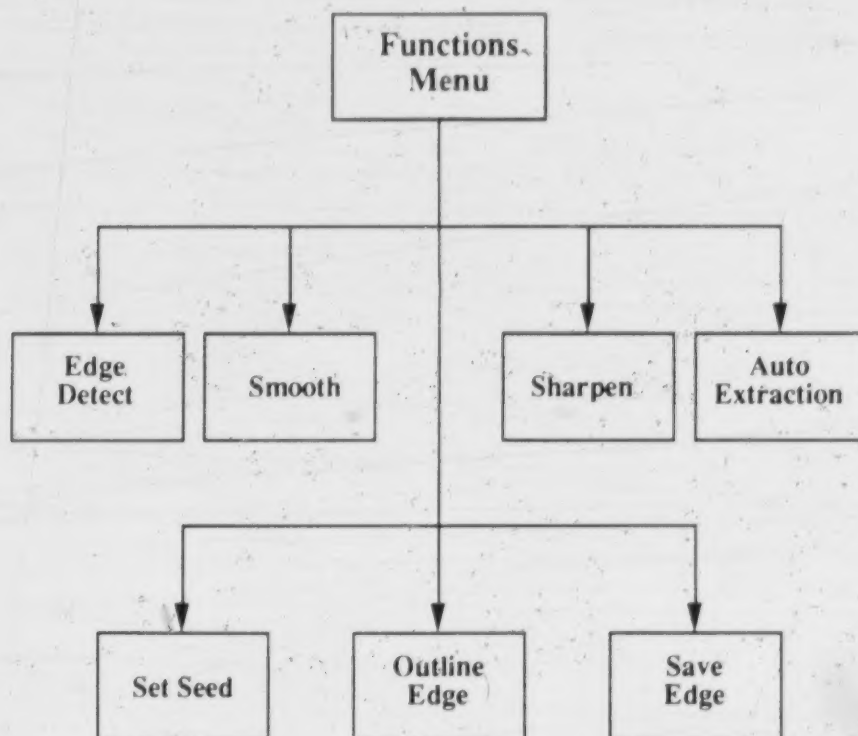


Figure 4.19: Structure of Functions Menu.

#### 1) Edge Detect( command-T )

This function produces a binary image using the Prewitt operator as described in Section 3.7. Two convolutions are done, generating vertical and horizontal gradients and then thresholding is applied to produce a binary image. A dialog box is invoked to let the user to enter a threshold value before the operation applied to the image.

#### 2) Smooth ( command-C )

This function applies neighbourhood average technique discussed in Section 3.6.2 to the image. This filter blurs (softens) the image.

#### 3) Sharpen ( command-B )

This function provides unsharp masking technique discussed in Section 3.6.3 to the image. This filter increases the contrast and accentuates detail of the image, but it may also accentuate noise.

#### 4) Set Seed ( command-F )

This function allows the user to select a discontinuous boundary portion by setting a seed in the boundary region or to the right of the boundary region. The selected seed is used in tracing the edges.

#### 5) Outline Edge ( command-V )

This function extracts and displays the boundary selected in the Set Seed function. The extracted boundary data is temporarily stored in a buffer area before writing to a file. It allows the user to preview the result of edge tracing.

#### 6) Save Edge ( command-S )

This function saves the boundary data in a buffer to a file. If it is the first file in a set of magnetic resonance images, the user is asked to enter a file name by the standard file dialog box; otherwise, the data will be stored to the previously opened file.

#### 7) Auto-Extraction ( command-A )

This function performs the contour extraction step discussed in Section 3.4.4. If it is the image in an image data set, a dialog box is displayed for the user to enter a boundary data file name. If auto-processing is selected in NEW SET Command, the extracted boundary will be displayed and a dialog box is invoked to let the user select whether the data will be saved. Then the next image file is read and the same sequence of filtering operations applied to the first image file will be automatically applied to this file. The same operating process is applied to the rest of the image files in the image data set.

### 4.3 '3D' - Reconstruction Software

'3D' is the application for reconstruction of the boundary data extracted from the program 'HEAD'. It requires two megabytes of memory to execute. After selecting the boundary data file, the boundary data is read in to create a three-dimensional model having the data structure as described in Section 3.8. Then a window with six control scrollbars is created for the displaying of the three-dimensional model. The control scrollbars allows the user to rotate the three-dimensional model about x,y, z axis, zoom in and out on the model and translate the three-dimensional model upward/downward or left/right.

'3D' has the same event loop structure as 'HEAD' and is shown in Figure 4.20.

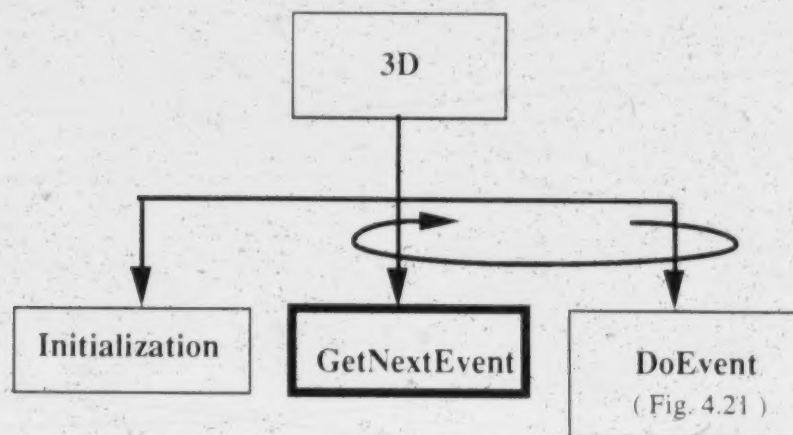


Figure 4.20: Structure chart of 3D.

#### 4.3.1 Descriptions of Software Routines

The following describes the function of each subroutine implemented in the program '3D'.

##### 1. DoEvent (Figure 4.20 and 4.21)

This function determines what kind of event from the **event->what** field. If it is a **mouseDown** event, the location of the mouse click is determined from the **event->where** field. Then it calls the appropriate routines to handle the user-initiated event.

## 2. DoMenuCommand (Figure 4.21 and 4.22)

This function determines which menu item is selected by examining the higher-order and lower-order word of the **event->where** field. Then it passes control to the corresponding routines to handle the user request.

## 3. DoUpdate (Figure 4.21 and 4.23)

This function updates the contents of the currently active window by calling subroutine *DrawWindow*.

## 4. ContentClick (Figure 4.21)

This function is called when a **mouseDown** occurs in the region of a window. If the mouse click position is in the viewing region of the window, no action is performed; otherwise, it determines which control scrollbar and what control action are activated and then executes the appropriate code.

## 5. DoActivate (Figure 4.21)

This function is called when the user switches between applications. If the application window is the front-most window, it calls subroutine *DoUpdate* to redraw the window.

## 6. DisplayModel (Figure 4.21 and 4.24)

This function displays the three-dimensional model with or without hidden surface removal. It makes use of the Toolbox routine **MoveTo3D** and **LineTo3D**.

#### 7. Create3DWindow (Figure 4.22)

This function creates a new window by reading the window parameters from the resource file. After the window is created, it creates the six control scrollbars and a offscreen grafport for the three-dimensional display of the model.

#### 8. OpenFile (Figure 4.22)

This function uses a standard file dialog box to ask the user to select the boundary data file to open. Then it reads in the boundary data, creates a three-dimensional model, and displays the model using the same orientation as in 'HEAD'.

#### 9. DoCloseWindow (Figure 4.22)

This function closes the active window and disposes all memory space requested by the program. Then it resets the graf3D parameters.

#### 10. Terminate (Figure 4.22)

This function quits the execution of the program. Before returning to MultiFinder, it closes the active window and releases all of the memory space originally requested by this application.

#### 11. ReadFile (Figure 4.22)

This function reads the boundary data for reconstruction purpose. It first reads in the header of the boundary data file to extract parameters and size of the file. Then it reads in the boundary data and creates a three-dimensional model as described in Section 3.8.

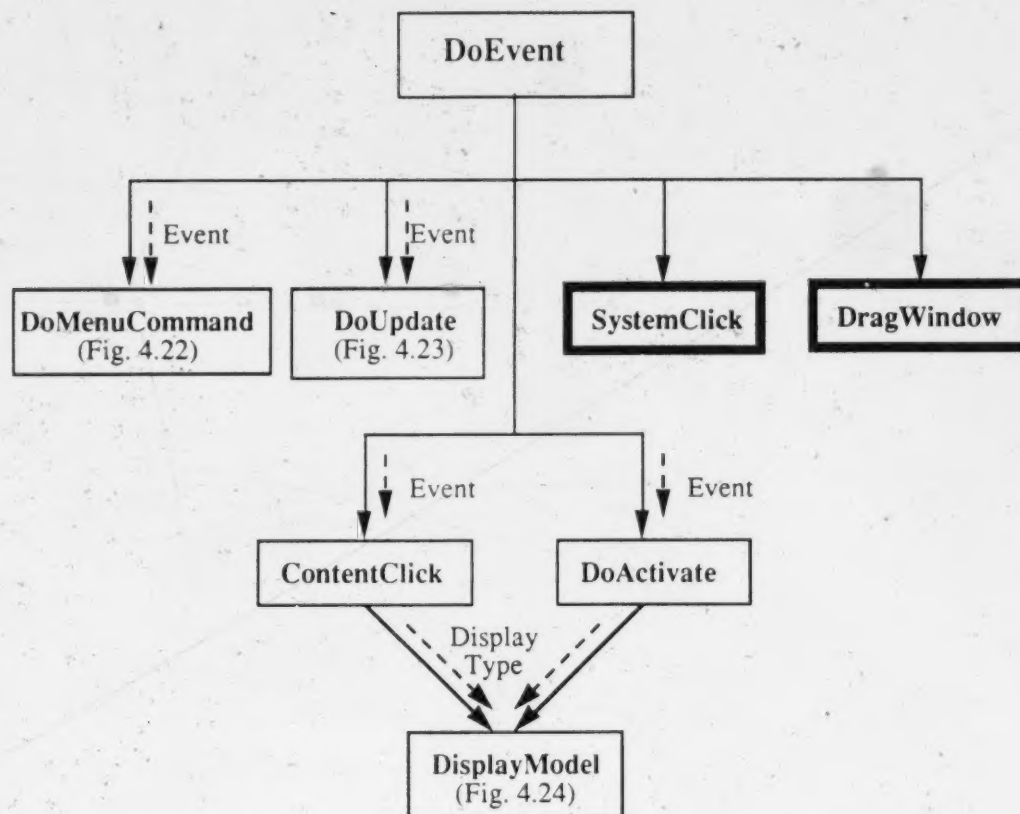


**Figure 4.21:** Structure chart of DoEvent.

## 12. ReadFile (Figure 4.22)

This function reads in the boundary data for reconstruction purpose. It first reads in the header of the boundary data file to extract parameters and size of the file. Then it reads in the boundary data and creates a three-dimensional model as described in Section 3.8.

## 13. CreateACell (Figure 4.22)

This function dynamically allocates memory space and reset all the fields for a data record.

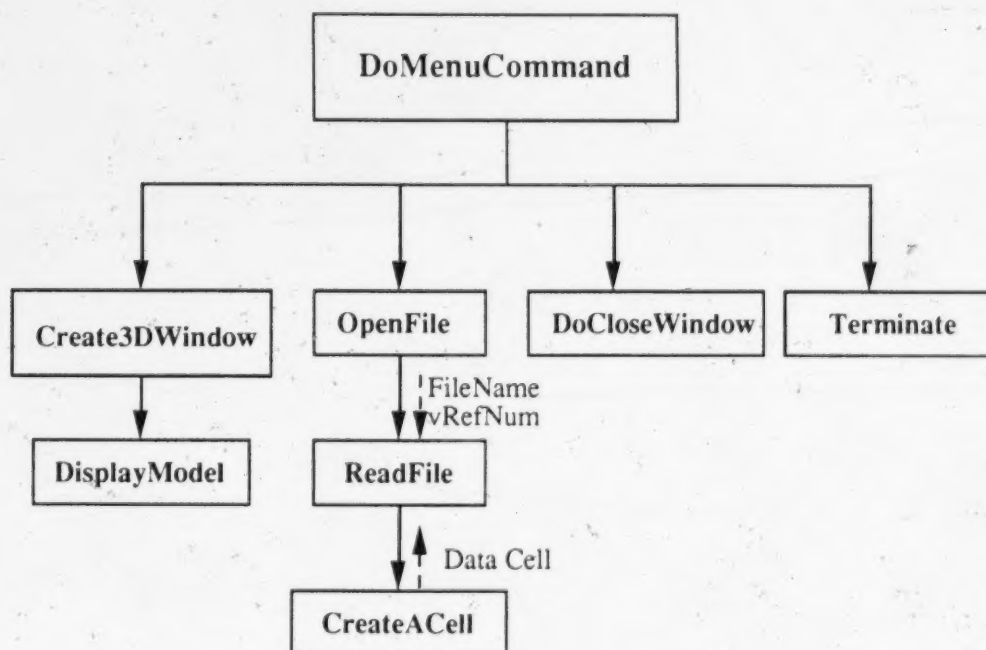


Figure 4.22: Structure chart of DoMenuCommand

14. DrawWindow (Figure 4.23)

This function redraws the control scrollbars of the window and displays the three-dimensional model of the skull.

15. RedrawControls (Figure 4.23)

This function redraw the control scrollbars and displays the labels corresponding to the control scrollbars by calling subroutines *DrawControls* and *PrintLabels*.

16. DisplayModel (Figure 4.23 and 4.24)

This function displays the three-dimensional model of the skull as a wire-frame with or without hidden surface removal according to the passing parameter.

17. PrintLabels (Figure 4.23)

This function is used to draw the labels for each control scrollbars. The directions of the rotation are shown at each end of the scrollbars and the titles of the scrollbars are displayed in the middle of the scrollbars. Two rectangular frames per scrollbar are also drawn in which the degree of rotation is displayed inside the rectangle.

18. AdjustDisplay (Figure 4.23)

This function updates the degree of rotation in the control scrollbar by drawing the thumb of the scrollbars in the proportional position.

19. WireDisplay (Figure 4.24)

This function displays the three-dimensional model as a wire-frame without hidden surface removal. It transverses the three-dimensional model array and draws lines between data points by using Toolbox routines **MoveTo3D** and **LineTo3D**.

20. HiddenLineDisplay (Figure 4.24)

This function displays the three-dimensional model in wire-frame mode with hidden surface removal. It sets the visibility flag of the data cell by calling subroutine *SetDisplay*. After setting the visibility flag, it transverses through the three-dimensional model array and display the model using Toolbox routines **MoveTo3D** and **Lineto3D**.

21. SetDisplay (Figure 4.24)

This function sets the visibility flag of the three-dimensional model using hidden surface algorithm.

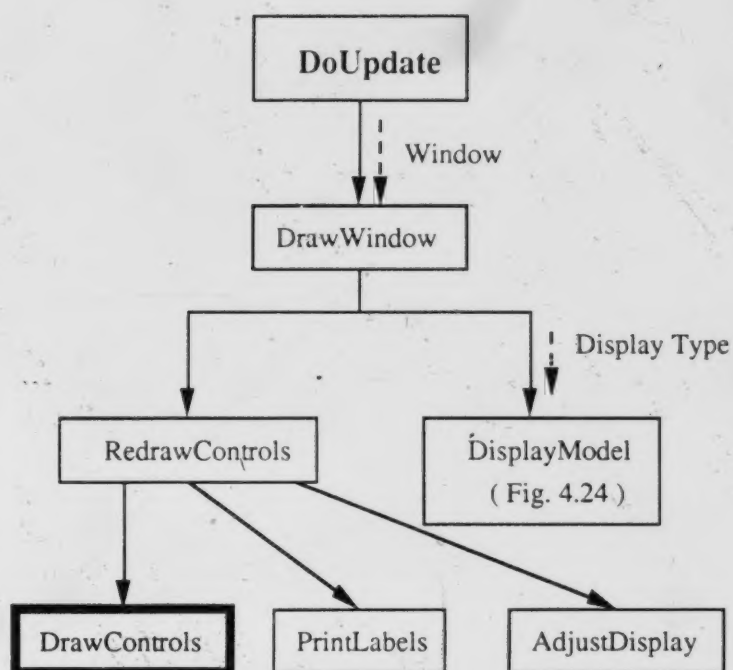


**Figure 4.23:** Structure chart of `DoUpdate`.

## 22. CheckVisibility (Figure 4.24)

This function determines the visibility of data points in the three-dimensional model. It checks whether the data point is within the region of a surface. If it is within the region of a surface, it compares the  $z$  value of the surface at that location to the  $z$  value of the data point. If the  $z$  value of the data point is larger, then the data point is visible and the visibility flag is set.

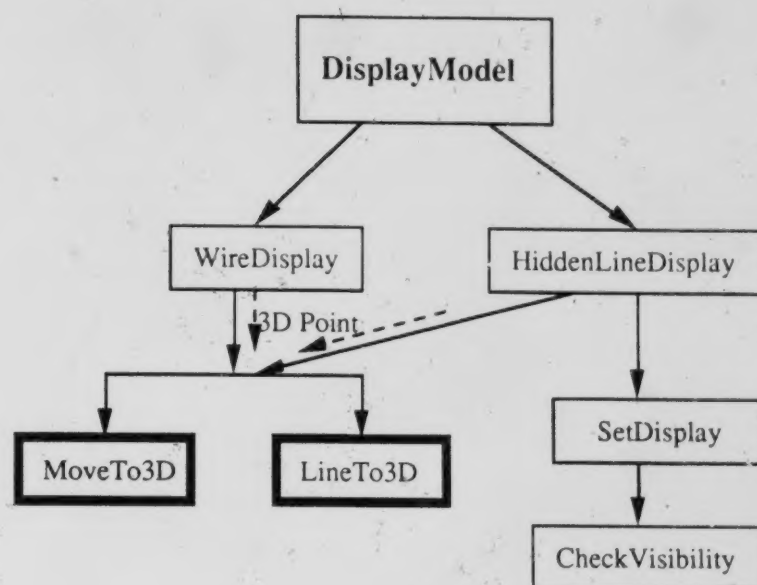


**Figure 4.24:** Structure chart of DisplayModel.

### 4.3.2 Menu Command

'3D' has two menu bars: File Menu and Display Menu. Each menu item is assigned a command-key for quick selection instead of using the mouse to pull down the menu bar all the time.

#### File Menu

The File menu includes operations to manipulate the input 3D data files as described in Section 3.38 and 3.39. Its structure is shown in Figure 4.25.

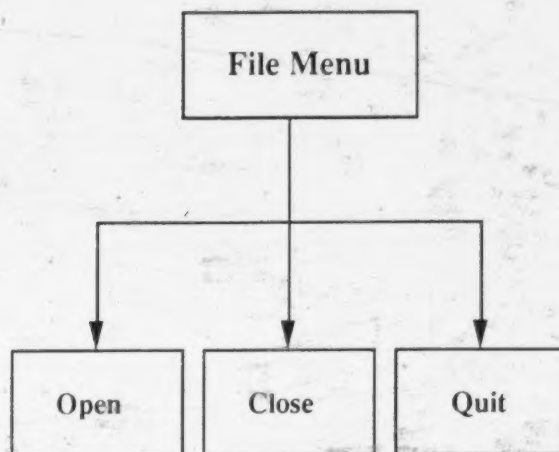


Figure 4.25: Structure of File Menu.

**1) Open ( command-O )**

This function opens the 3D data file created by 'HEAD' and creates a three-dimensional model. It displays a standard file dialog box for the user to select the file to open. Then it reads the boundary data for the reconstruction process. After finishing the reconstruction process, the three-dimensional model is displayed in a window having the same orientation as displayed in 'HEAD'.

**2) Close ( command-W )**

This function closes the current three-dimensional display window.

### 3) Quit ( command-Q )

This function closes the current 3D window, stops the execution of the application, and returns control to MultiFinder.

### Display Menu

The Display menu allows the user to select the wire frame display with or without hidden line removal (as described in Section 3.10 and 3.11). Its structure is shown in Figure 4.26.

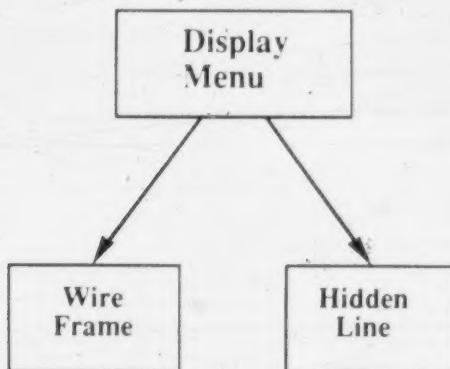


Figure 4.26: Structure of Display Menu.

#### 1) Wire Frame ( command-F )

This function displays the 3D model without hidden line removal.

#### 2) Hidden Line ( command-H )

This function displays the 3D model with hidden line removal.

### References

- [1] Apple Comp. Inc. Inside Macintosh Vol/ 1 to 5, Addison-Wesley Publishing Company, Inc., July 1988.
- [2] Apple Comp. Inc. Programmer's Introduction to the Macintosh Family, Addison-Wesley Publishing Company, Inc., 1988.
- [3] Sommerville I. Software Engineering, 2nd Edition, Addison-Wesley Publishing Company, Inc., 1985.

## CHAPTER 5: SAMPLE APPLICATION SESSION

The following pages illustrate the use of 'HEAD' and '3D' with a set of magnetic resonance input images (image data set). Before starting the sample session, the imaging parameters (15 slices each with a slice thickness of 5 millimetres, an inter-slice factor of 1.2 and 250 millimetres over 256 pixels) are obtained from the parameter print-out provided with the magnetic resonance images. The three-dimensional reconstruction process involves two steps: boundary extraction and three-dimensional model creation. The program HEAD performs the first step. It reads the image data set, performs enhancement operations on the images if needed, extracts the boundary data and stores the boundary data on a file (referred to as a boundary data file). The program 3D reads the boundary data file, uses the boundary data to create a three-dimensional model, and displays the model in wire-frame mode with or without hidden surface removal.

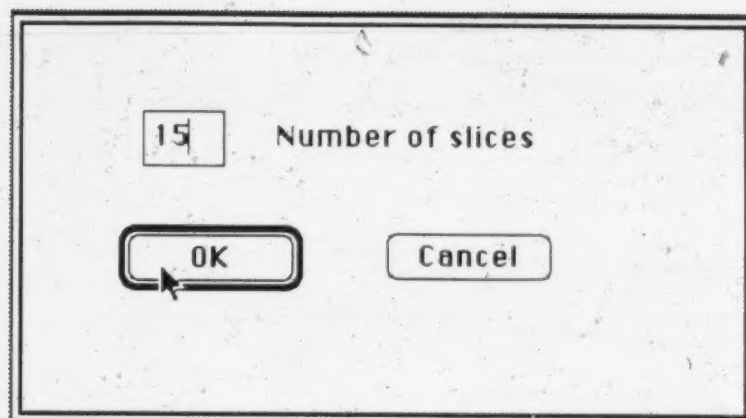
### 5.1 'HEAD'

HEAD is invoked by double clicking on the application icon. Before opening the magnetic resonance image file, the command 'New Set', under File Menu, must be selected. This function displays four dialog boxes that request the user to input image parameters (shown in Figures 5.1 and 5.2). The first dialog box (Figure 5.1(a)) asks for the number of input slices; enter '15'. The second dialog box (Figure 5.1(b)) queries the user to select the type of input data source. Either magnetic resonance imaging or computer tomography data can be selected provided the input data files conform to the file format discussed in Section 3.4. This selection is required because magnetic resonance imaging data and computer tomography data require different calculations in pixel size and coordinates. Currently, only the magnetic resonance imaging calculation is

implemented in this application and CT data is not implemented. The third dialog box (Figure 5.2(a)) asks the user to enter the magnetic resonance imaging parameters. These parameters are available from the magnetic resonance imaging parameter files supplied by the magnetic resonance imaging unit. Default values are displayed in the dialog box and can be changed by moving the cursor to the data box and typing in the new values. The fourth dialog box (Figure 5.2(b)) selects the auto-process feature of the application. If the auto-process feature is selected, by clicking on the 'YES' box, the application will record the last five filtering operations applied to the first image file, and automatically apply this sequence to the files in the rest of the data set. After applying a sequence of operations, the extracted boundary is displayed and a dialog box is used to ask the user whether he/she wants to save the data. If the user selects to save the file, the extracted data will be saved and the next image file will be opened. The same processing steps are repeated until the end of the image data set is reached.

Activating the 'Open' command under File Menu opens an image file, and selects the slice suffixed with .1. Two identical images are displayed on the screen as shown in Figure 5.3. The window on the left-hand side is the working window on which the editing and filtering operations are performed and displayed. The window in the right-hand side is the reference window showing the original image.

The 'Smooth' command under the Functions Menu is selected to removed low gray-level variation in the background, especially in the middle section of the image. Then the 'Trace Edge' command under the Functions Menu is used to produce a binary map. A dialog box is displayed to let the user to enter a threshold value for the edge detection algorithm as discussed in Section 3.5. The result of applying 'Smooth' and 'Edge Detect' is shown in Figure 5.4.

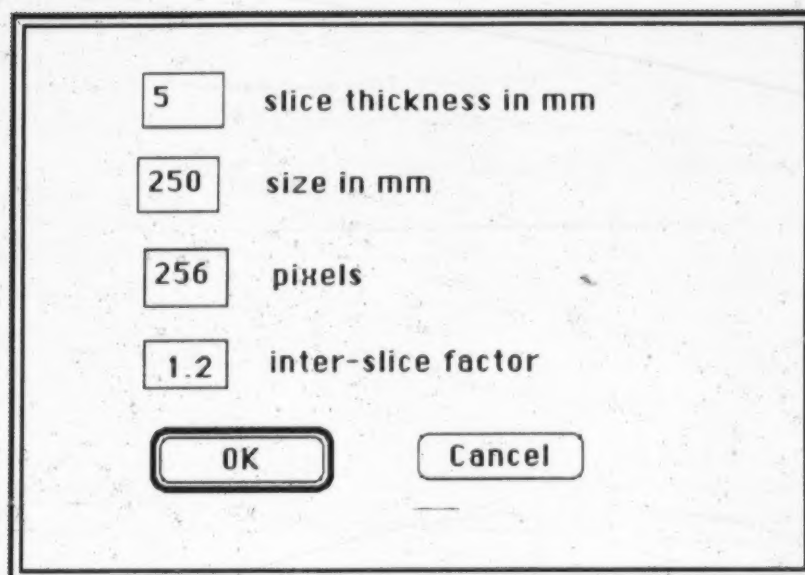


(a)



(b)

Figure 5.1: Dialog boxes for 'New Set' Menu Item.



5 slice thickness in mm

250 size in mm

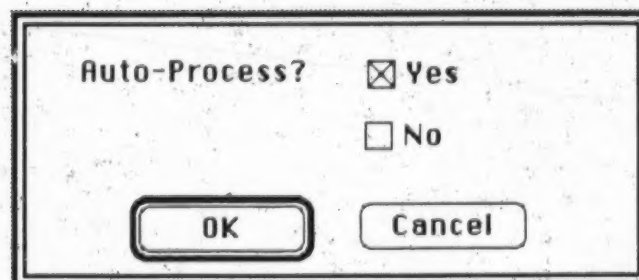
256 pixels

1.2 inter-slice factor

OK Cancel

This dialog box contains four input fields with their respective labels: '5' for 'slice thickness in mm', '250' for 'size in mm', '256' for 'pixels', and '1.2' for 'inter-slice factor'. At the bottom, there are two buttons: 'OK' and 'Cancel'.

(a)



Auto-Process? ☒ Yes  
☐ No

OK Cancel

This dialog box has a label 'Auto-Process?' followed by two radio button options: 'Yes' (which is selected) and 'No'. At the bottom, there are two buttons: 'OK' and 'Cancel'.

(b)

Figure 5.2: Dialog boxes for 'New Set' Menu Item.



Figure 5.3: Start-up display after reading in the first file.

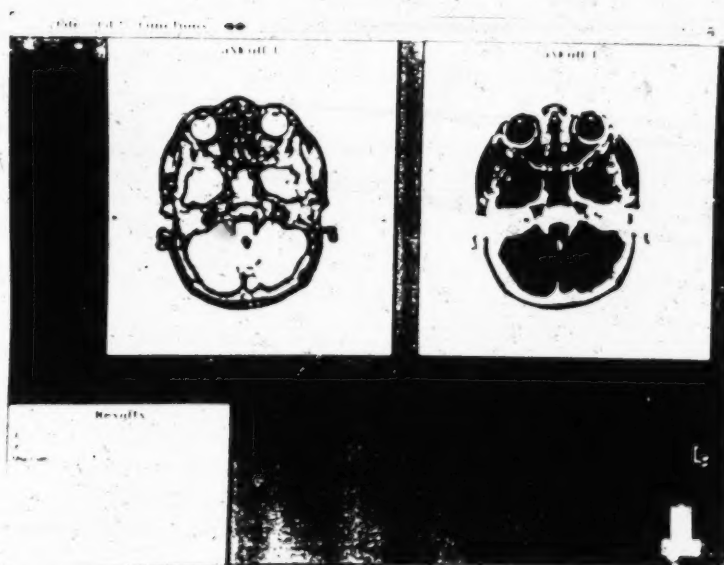


Figure 5.4: Result of applying 'Smooth' and 'Edge Detect' to the first image of a set of magnetic resonance images.

The 'Auto-Extraction' command under Functions Menu is selected to trace the boundary of the skull. A standard file dialog box is displayed to ask the user to enter the file name, type in 'Data1'. After tracing the boundary of the skull, the boundary shape is displayed and a dialog box is used to ask the user to save the data. If the user selects to save the file, the data is stored in the given file name by the user. Then, the next image file will be opened and the same sequence of operations: 'Smooth' followed by 'Edge Detect' will be applied to the image. The same processing step is repeated until the end of the image data set is reached. If the user wants to manually intervene in the filtering operations of any image during the auto-process step, he/she can select the 'No' options in the save file dialog box. The extracted data will be discarded and the original read-in image is displayed in the working window. The user can select any sequence of filtering or editing operations for the image and can save the boundary data. After the data is saved, the process returns to the auto-process state and the next file is opened. Then the original sequence of operations is applied to the new image. After going through 15 images in this set of magnetic resonance imaging scans, the boundary data is stored in a file. If the user wants to input the next set of image files, he/she needs to select the 'New Set' command and repeats the procedure described above. The user can quit the program by select 'Quit Head' under File menu or command-Q.

## 5.2 '3D'

Three dimensional reconstruction is performed by 3D. After the application is invoked by double clicking on the application icon, the 'Open' command, selected under File Menu, opens a boundary data file. A dialog box is displayed to list all the boundary data files created by HEAD. After the user selects the boundary data file by double clicking on the name list, the boundary data is read in to create a three-dimensional model as described in Section 3.8. This reconstruction process takes about 3 minutes. Then the

three-dimensional model is displayed as a wire-frame model on the screen with the same orientation as displayed in HEAD, without hidden line removal (shown in Figure 5.5). The user can switch between wire-frame display with or without hidden line removal by selecting the menu items 'Wire Frame' or 'Hidden Line' under Display Menu.

Rotation about the x, y, or z axis is activated by clicking on the appropriate arrow, clicking in the gray region or by moving the thumb in the control scrollbars. The amount of rotation is shown in the display box under each of the control scrollbars. These values are measured from the initial position of the head when the file was opened. Three instances of the rotation with and without hidden line removal are shown in Figure 5.6 to Figure 5.8. The zooming feature is activated by clicking in the gray region in the zoom control scrollbar. Figure 5.9 shows the result of zooming in the 3D model. Translation can also be activated a similar way as rotation. The user can quit the program by selecting 'QUIT' under File menu or command-Q.

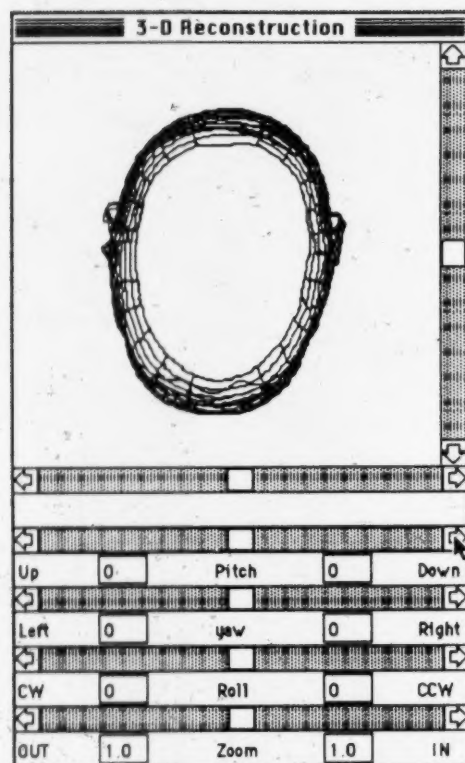


Figure 5.5: Preliminary '3D' display.

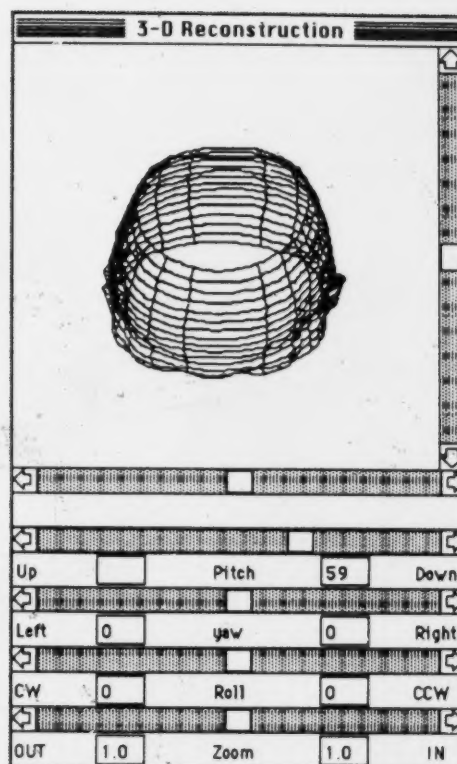


**Figure 5.6:** Rotation of the 3D model about x axis  
without hidden surface removal.

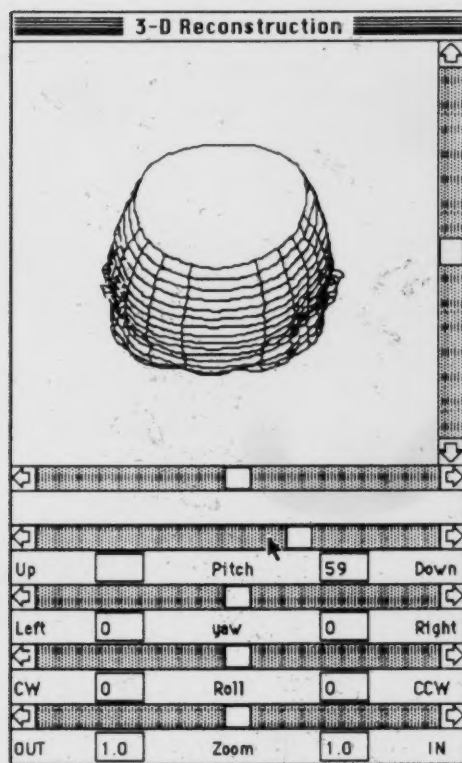


**Figure 5.7:** Rotation of the 3D model about x axis  
with hidden surface removal.

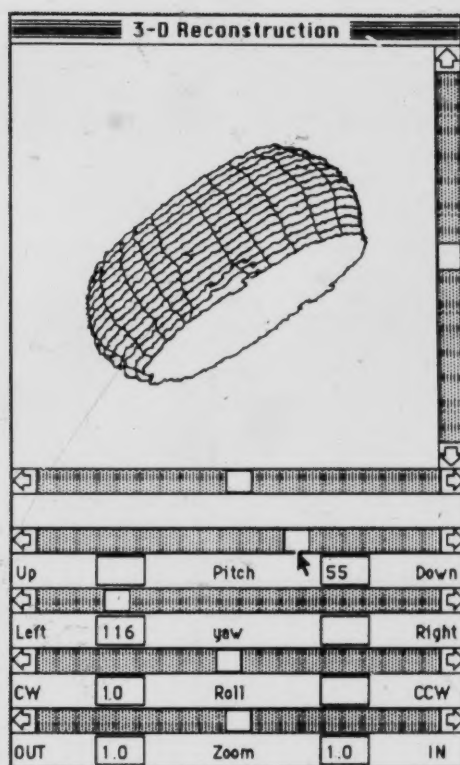


Figure 5.8: Rotation of the 3D model about x, y and z axis with hidden surface removal.

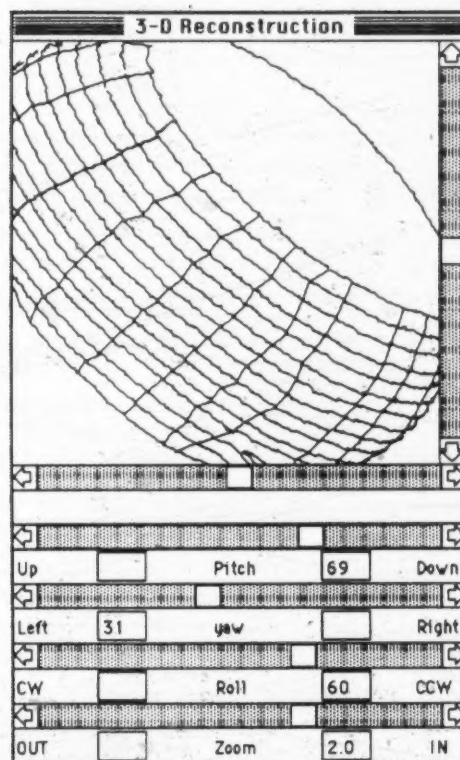


Figure 5.9: Zooming in of the 3D model.

## CHAPTER 6: CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

### 6.1 Conclusion

The purpose of this thesis was to utilize image processing techniques to perform three-dimensional reconstruction of the skull from magnetic resonance images. The software application was implemented by two programs: 'HEAD' and '3D'. 'HEAD' reads the magnetic resonance image data set, extracts the boundary data of the skull, and stores the boundary data in a file. '3D' reads the boundary data produced by 'HEAD' as input, uses these data to create a three-dimensional model, displays the three-dimensional model in wire-frame mode with or without hidden surface removal, and provides controls for rotation, zooming, and translation of the three-dimensional model.

A three-dimensional reconstruction application was developed to run on inexpensive computer hardware and to provide a user friendly interface. It is also easy to use and allows the user to observe the shape of a skull in a clinical environment. The application is planned to be used in the study of scaphocephaly by the Rehabilitation Engineering Department of Glenrose Rehabilitation Hospital described in Chapter 1. This study will investigate the effect of surgical intervention on scaphocephaly by comparing the skull shape data collected from children with scaphocephaly to those from the normal children and will try to find out an objective method for measuring the effectiveness of the surgical treatment.

## 6.2 Limitations

The application was developed on a Macintosh II computer running MultiFinder. Interactions between separate applications running at the same time and between applications and desktop accessories are not always predictable and this caused major problems in software development. Also, the development tools contain bugs which cause them to quit without warning. This was compounded by the lack of expert knowledge regarding Macintosh programming, especially in the areas of managing the window environment and the graphical user interface.

The application is not easily portable from the Macintosh computer to other machines because of the use of machine-specific Toolbox routines for graphical display. Also, there is a problem with the speed of the application when floating-point arithmetic is used. Even though the three-dimensional model representation and transformation calculations are implemented in FIXED (four-byte integer) type, the tile-line-creation requires the use of floating-point arithmetic which greatly reduces the speed of the program. It takes about 30 seconds to create the three-dimensional model from 15 image files but takes about 2 minutes to create the tile lines for the model. This time factor can be improved by using specialized graphics hardware such as dedicated floating-point array processors and graphics accelerators, but then the purpose of running the application on a general desktop computer is not achieved.

The image enhancement and boundary extraction functions are aimed at the outside skull boundary. If both the inside and outside boundaries of the skull is required in the future, modifications are needed to accomplish the requirements. Also, the application only deals with magnetic resonance image data and modifications are required to import computed tomography data as well.

### 6.3 Recommendations for Future Work

Wire-frame display is the most basic method for a three-dimensional surface representation. It gives the general shape of the three-dimensional object but lacks the detailed information on the surface of the object. Further development should investigate the use of shading methods that provide a realistic appearance to the reconstructed structure. The simplest shading method is distance shading which assigns a gray level to the surface proportional to the distance from the surface to the viewer. This method is very computationally efficient because the complex process of estimating surface normals is not involved. Two other common shading methods are Gouraud shading[4] and Phong shading[7]. These two methods both utilize the normal vector of the surface; therefore, they are computationally expensive and specialized graphics accelerator or parallel processing is required to attain a reasonable speed in executing the program. For a short term development, shading can be achieved by using an existing software package for manipulating and displaying three-dimensional data such as 3D Graphic Tools by Micro System Options[1].

Further investigation is also required into three-dimensional surface representation methods. A number of surface representation models have been proposed in the literature and the three of them are as follows:

1. The Cuberille Model proposed by Herman and Liu[5] partitions the image space into unit cubes by three orthogonal sets of equally spaced parallel planes. The volume elements become components of a cuberille  $C$  and all surface elements are faces of  $C$ .
2. The Triangular Tile Model represents a three-dimensional surface by means of polygonal patches. The most popular and simplest of the polygons to represent the surface of the reconstructed structure is the triangle. Keppel[6], Fuchs

*et al.*[13] and Christiansen *et al.*[2] proposed three different approaches making use of the triangular tile technique.

3. Wu, Abel and Greenberg[8] proposed a technique which reconstructs surfaces with a B-spline approximation interpolated between the contour planes.

Short-term development can also emphasize improvement in the application features. For example, the "Auto-Extraction" function in 'HEAD' only extracts the outer surface of a skull with an image size of 256x256. The extraction capability can be improved by using the mouse to select a region of interest in the image. This feature would permit the three-dimensional reconstruction of objects within the skull, but problems of data registration need to be addressed. In addition, the 'PRINT', 'CUT', 'COPY' and 'PASTE' functions can enhance the analysis ability of the application.

In conclusion, this project has been a practical and rewarding experience with much still to be learned in three-dimensional image processing and Macintosh programming.

## References

- [1] Apple Comp. Inc. "APDAlog", Spring 1990.
- [2] Christiansen H.N. Sederberg T.W., "Conversion of Complex Contour Line Definitions into Polygonal Element Mosaics", Computer Graphics, Vol. 12, Aug. 1978, pp. 187-192.
- [3] Fuchs H. Kedem Z.M. & Uselton S.P., "Optimal Surface Reconstruction from Planar Contours", Comm. ACM, Vol. 20, Oct. 1977, pp. 693-702.
- [4] Gouraud H. "Computer Display of Curved Surfaces", IEEE Trans. Computers, Vol. C-20, June 1971, pp. 623-629.
- [5] Herman G.T. Liu H.K., "Three-Dimensional Display of Human Organs from Computer Tomograms", Comp. Graphics and Image Processing, Vol. 9, 1979, pp. 1-21.
- [6] Keppel E. "Approximating Complex Surfaces by Triangulation of Contour Lines", IBM J. Res. Development, Vol. 19, Jan. 1975, pp. 1-96.
- [7] Phong B.T. "Illumination for Computer Generated Pictures", Comm. ACM, Vol. 18, June 1975, pp. 311-317.
- [8] Wu S.H. Abel J.F. & Greenberg D.P., "An Interactive Computer Graphics Approach to Surface Representation", Comm. ACM, Vol. 20, Oct. 1977, pp. 703-712.